НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут телекомунікаційних систем

Кафедра Телекомунікаційних систем

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри
_____ Л.О. Уривський
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 172 Телекомунікації та радіотехніка

на тему: «Дослідження систем аналізу великих масивів неструктурованих даних»

Виконав:
студент ІІ курсу, групи ТС-61м
Бабич Микола Валерійович _____

Керівник:
ст. викладач кафедри ТС
Лісковський Ігор Олегович _____

Рецензент:
Посада, науковий ступінь, вчене звання,
Прізвище, ініціали _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2018 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут телекомунікаційних систем

Кафедра Телекомунікаційних систем

Рівень вищої освіти – другий (магістерський) за освітньо-науковою програмою

Спеціальність (спеціалізація) – 172 «Телекомунікації та радіотехніка» (172.3620.1 «Телекомунікаційні системи та мережі»)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Л.О. Уривський

«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Бабичу Миколі Валерійовичу

1. Тема дисертації «Дослідження систем аналізу великих масивів неструктурованих даних», науковий керівник дисертації Лісковський Ігор Олегович ст. викладач кафедри ТС, затверджені наказом по університету від «06» квітня 2018 р. №1105-с

2. Термін подання студентом дисертації _____

3. Об’єкт дослідження неструктурована інформація.

4. Предмет дослідження алгоритми аналізу неструктурованої інформації.

5. Перелік завдань, які потрібно розробити:

- розглянути що таке Big Data, неструктуровані дані і чому важлива обробка неструктурованих даних;

- розглянути алгоритми аналізу неструктурованої та слабоструктурованої інформації;

- визначити критерії оцінки алгоритмів аналізу неструктурованої та слабоструктурованої інформації;

- дослідити оцінки алгоритми аналізу неструктурованої та слабоструктурованої інформації;

- розробити класифікатор тексту за допомогою програмного забезпечення SPSS Modeler;

- протестувати моделі в середовищі SPSS Modeler.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу

Плакат №1 «Тема, мета та завдання магістерської дисертації»

Плакат №2 «Актуальність та постановка задачі»

Плакат №3 «Області застосування Big Data»

Плакат №4 «Алгоритми аналізу неструктурованих даних»

Плакат №5. «Порівняння алгоритмів аналізу неструктурованих даних»

Плакат №6. «Моделювання класифікатору тексту»

Плакат №7. «Результати моделювання розробленого класифікатора»

Плакат №8. «Висновки»

7. Орієнтовний перелік публікацій

Бабич М.В. Алгоритм аналізу неструктурованої та слабо структурованої інформації / Бабич М.В. // Міжнародна науково-практична конференція «Наука та освіта: ключові питання сучасності»: зб. наук. праць «λόгοσ». – 2018.

8. Дата видачі завдання 10 вересня 2016 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Розробка, оформлення, узгодження та затвердження технічного завдання на роботу. Аналітичний огляд інформаційних матеріалів. Підбір та опрацювання необхідної науково-технічної літератури.	01.09.2016- 31.12.2016	
2	Огляд областей застосування Big Data та вивчення основної термінології Big Data.	10.01.2017 - 29.02.2017	
3	Розгляд алгоритмів аналізу неструктурованої та слабо структурованої інформації.	01.03.2017 – 30.07.2017	
4	Пропозиції щодо алгоритмів аналізу неструктурованої та слабо структурованої інформації.	01.08.2017 – 31.10.2017	
5	Дослідження алгоритмів аналізу неструктурованої та слабо структурованої інформації.	01.11.2017 – 30.01.2018	
6	Розробка класифікатора текстів.	01.02.2018 – 31.03.2018	
7	Реалізація класифікатора текстів за допомогою ПЗ SPSS Modeler.	01.04.2018 – 30.04.2018	
8	Узагальнення і оцінювання результатів досліджень, підготовка підсумкового звіту. Подання роботи до приймання, та її захист.	01.05.2018 - 20.05.2018	

Студент Бабич М.В.

Науковий керівник дисертації Лісковський І.О.

РЕФЕРАТ

Обсяг магістерської дисертації складає 85 сторінок, зокрема 20 ілюстрації, 14 таблицю, 6 формул та … джерело інформації.

Актуальність теми. За думкою експертів, більше ніж 85% даних формуються у неструктурованій формі. До неструктурованих даних можна віднести текст, мультимедія (відео, голос, зображення), тобто це дані, які не мають заздалегідь визначеної структури, або не організована у встановленому порядку. Це все призводить до труднощів аналізу, особливо у випадку використання традиційного програмного забезпечення, яке призначене для роботи зі структурованими даними. Повсякчас, у неструктурованих даних можливо знайти більш цікаві та потенційно більш ціні експертні оцінки, висновки. Особливо, якщо брати до уваги бурхливих розвиток гаджетів і IoT, які формують величезні потоки трафіку, які необхідно обробити.

Тема магістерської дисертації є актуальною, тому що незважаючи на бурхливий розвиток Big Data, технологія тільки на початку свого розвитку і з кожним роком кількість питань тільки збільшується. В обробці неструктурованих даних, перш за все зацікавлений бізнес, тому що з розвитком соціальних мереж кількість актуального неструктурованого контенту зростає і обробляти інформацію вручну не під силу. Саме тому перед дослідниками постає питання автоматичної обробки та класифікації тексту.

Метою випускної кваліфікованої роботи є дослідження алгоритмів аналізу неструктурованої та слабо структурованої інформації та створення класифікатору тексту за допомогою середовища моделювання SPSS Modeler.

Відповідно до поставленої мети були сформульовані такі *завдання*:

– розглянути основні алгоритми аналізу неструктурованої та слабоструктурованої інформації;

- виробити критерії оцінки алгоритмів неструктурованої та слабоструктурованої інформації;
- розробити класифікатор тексту на основі процесу CRISP-DM;
- реалізувати класифікатор тексту у середовищі моделювання SPSS Modeler;
- протестувати роботу моделі.

Об'єктом дослідження є неструктуровані дані.

Предметом дослідження є алгоритми аналізу неструктурованої та слабоструктурованої інформаціх.

Методи дослідження. В ході роботи були використані: методи теоретичного дослідження, емпіричний підхід, методи логічного проектування та процедурної алгоритмізації, прийоми динамічного програмування.

Апробація результатів дисертації. Основні результати дисертаційного дослідження оприлюднено в ході міжнародної науково-практичної конференції «Наука та освіта: ключові питання сучасності», 2018. (м. Чернігів)

Публікації. Основні положення і результати дисертаційної роботи знайшли своє відображення на Міжнародній науково-практичній конференції «Наука та освіта: ключові питання сучасності», 2018. (м. Чернігів).

Ключові слова: великі дані, неструктуровані дані, класифікатор тексту, SPSS Modeler, генетичний алгоритм.

ABSTRACT

The work contains 85 pages, 20 illustrations, 14 table, 6 formulas and … sources.

Relevance of the topic. According to experts, more than 85% of data is formed in an unstructured form. Unstructured data can include text, multimedia (video, voice, image), it is data that does not have a predefined structure, or not organized in the prescribed manner. All this leads to difficulty in analysis, especially when using traditional software that is designed to work with structured data. At all times, it is possible to find more interesting and potentially more valuable expert assessments, conclusions in unstructured data. Especially we see the rapid development of gadgets and IoT that form huge traffic flows that need to be processed.

The topic of the master's thesis is relevant, because despite the rapid development of Big Data, technology is only at the beginning of its development, and with each passing year the number of questions only increases. In the processing of unstructured data, primarily interested in business, as the development of social networks the amount of actual unstructured content increases and handled manual information is not feasible. That is why researchers are faced with the question of automatic processing and classification of the text.

The *purpose* of the thesis is to develop the algorithms for analysis of unstructured and poorly structured information and to create a text classifier using the SPSS Modeler simulation environment.

In accordance with the stated goal, the following *objectives* were formulated:

- to consider the basic algorithms of analysis of unstructured and poorly structured data;

- to develop criteria for evaluating algorithms of unstructured and poorly structured data;

- develop a text classifier based on the CRISP-DM process;

- to implement the text classifier in the SPSS Modeler simulation environment;

- test the model work.

The *object* of research is an unstructured data.

The *subject* of research are algorithms for analysis of unstructured and poorly structured data.

Research methods. In the course of the work were used: methods of theoretical research, empirical approach, methods of logical design and procedural algorithmization, techniques of dynamic programming.

Approbation of the results of the dissertation. The main results of the dissertation research was published during the iternational scientific and practical conference "Science and education: key issues of our time", 2018. (Chernihiv)

Keywords: Big data, unstructured data, text classifier, SPSS Modeler, genetic algorithm.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

AAPOR	(American Association for Public Opinion Research) - Американська асоціація дослідників громадської думки
BD	(Big Data) – великі дані
BoW	(Bag-of-Words) - це модель часто використовується при обробці текстів
CFS	Алгоритм кластеризації великих даних
CRISP-DM	(Cross Industry Standart Process for Data Mining) - міжгалузевий стандартний процес для дослідження даних
DT	(Decision Tree) – дерево рішень
DW	(Data Warehouse) – репозиторій даних
EMR	(Elastic MapReduce) – веб-сервіс, який обробляє великий об’єм даних
GPFS	(General Parallel File System) – загальна паралельна файлова система
HDFS	(Hadoop Distributed File System) – файлова система Hadoop
IG	(Information Gain) – приріст інформації
k-NN	(k-Naearest Neighbours) – метод k-ближчих сусідів
OLTP	(Online transaction processing) – обробка транзакцій в реальному часі
SMO	(Sequential Minimal Optimization) – метод оптимізації
SVM	(Support Vector Machines) – метод опорних векторів

ГА	Генетичні алгоритми
ГІС	Геоінформаційні системи
ССІ	Слабкоструктурована інформація
СУБД	Система управління базами даних

ВСТУП

З кожним роком кількість наукових праць, присвячених проблемам великих даних невпинно зростає. Основними джерелами вітчизняних дослідників є праці Л.Черняк, Н.В.Коритнікова, А.А.Берсегян, М.С.Купріянов, І. І. Холод, М. Д. Тесс, К.А.Хайдаров, І.А.Чубукова, А.А.Григорьєв, Н.А.Чалкіна, Н.Н.Двоєрядкіна. Серед зарубіжних видань найбільший внесок роблять компанії, які кваліфікуються на обробці великих даних, такі як IBM, Oracle та інші. Найбільш актуальну інформацію з приводу Big Data можна знайти в білих паперах зарубіжних компаній.

І тут виникає проблема в аналізі цих даних, тому що за інформацією експертів більше 85% даних зберігається в неструктурованій формі. До неструктурованих даних можна віднести текст, мультимедія (відео, голос, зображення), тобто це дані, які не мають заздалегідь визначеної структури, або не організована у встановленому порядку. Це все призводить до труднощів аналізу, особливо у випадку використання традиційного програмного забезпечення, яке призначене для роботи зі структурованими даними. Повсякчас, у неструктурованих даних можливо знайти більш цікаві та потенційно більш ціні експертні оцінки, висновки. Особливо, якщо брати до уваги бурхливих розвиток гаджетів і IoT, які формують величезні потоки трафіку, які необхідно обробити.

Тема магістерської дисертації є актуальною, тому що незважаючи на бурхливий розвиток Big Data, технологія тільки на початку свого розвитку і з кожним роком кількість питань тільки збільшується. В обробці неструктурованих даних, перш за все зацікавлений бізнес, тому що з розвитком соціальних мереж кількість актуального неструктурованого контенту зростає і обробляти інформацію вручну не під силу. Саме тому перед дослідниками постає питання автоматичної обробки та класифікації тексту.

Метою випускної кваліфікованої роботи є дослідження алгоритмів аналізу неструктурованої та слабо структурованої інформації та створення класифікатору тексту за допомогою середовища моделювання SPSS Modeler.

Найбільш зацікавленою галузю є продаж, тому що обробка неструктурованих даних, дає можливість працювати навпередження. І саме потреби бізнесу штовхають людство на освоєння технології Big Data та створення нових продуктів, які будуть вирішувати проблеми бізнесу пов’язані з аналізом великих даних.

РОЗДІЛ 1. ОГЛЯД ТЕХНОЛОГІЇ BIG DATA

1.1 Області застосування Big Data

Термін Big Data (BD) має безліч різних визначень і залишається досить розмитим поняттям. Можливо, через відсутність чіткого уявлення про сутність BD, всі можливості роботи в даній сфері не оцінюються до кінця належним чином.

На думку наукового редактора журналу «Відкриті системи. СУБД » Л. Черняка, остаточною датою народження Big Data можна вважати 3 вересня 2008 року. Це день, коли «вийшов спеціальний номер найстарішого британського наукового журналу Nature, присвячений пошуку відповіді на питання: як можуть вплинути на майбутнє науки технології, що відкривають можливості роботи з великими обсягами даних? Спеціальний номер підсумовує попередні дискусії про роль даних в науці взагалі і в електронній науці (e-science) зокрема »[1].

BD не просто технологія, це парадигмальний зсув, нова емпірична культура, яка практично формується в реальному часі, на наших очах. К. Лінч (редактор журналу Nature) «запропонував для нової парадигми спеціальну назву "Великі дані", вибране ним за аналогією з такими метафорами, як велика нафта, велика руда і т.п., що відображають не тільки кількість чогось, скільки перехід кількості в якість»[1].

Американська асоціація дослідників громадської думки (AAPOR) в своєму звіті про «Великі дані» [2] також розглядає BD як парадигмальний зсув в оглядовому/соціальному дослідженні: «За останні роки ми спостерігаємо збільшення обсягу статистичної інформації, яка описує різні соціальні феномени, засновані на так званих "Великих даних" ... Зміна в становленні нових типів даних, їх доступність, способи збору і поширення фундаментальні. Ця зміна підкреслює парадигмальний зсув в масових опитуваннях (survey researches) »[2, с. 9].

У цьому ж звіті наводиться досить загальне визначення BD: «Термін "Великі дані" – це свого роду опис великих за обсягом і різноманітних за складом характеристик, практик, технічних прийомів, етичних проблем і наслідків, які пов'язані з даними» [3, с. 14]. З посиланням на звіт компанії «Гартнер» автори розглядають основні характеристики «Великих даних» [2, с. 18]: для опису основних характеристик BD в англійській мові використовують три букви «V»: Volume (об'єм), Velocity (швидкість) і Variety (різноманіття).

Отже, стає очевидно, що Big Data - це комплексне поняття, що поєднує в собі:

1) безпосередньо дані (безліч закодованої інформації);

2) сукупність технологій роботи з цими даними;

3) новий погляд, нову парадигму в науці про дані (data science).

Джерел великих даних в сучасному світі безліч. У їх якості можуть виступати дані, які безперервно надходять з вимірювальних пристроїв, події від радіочастотних ідентифікаторів, потоки повідомлень з соціальних мереж, метеорологічні дані, дані дистанційного зондування землі, потоки даних про місцезнаходження абонентів мереж стільникового зв'язку, пристроїв аудіо- і відеореєстрації. Власне, масове поширення перерахованих вище технологій і принципово нових моделей використання різного роду пристроїв і інтернет-сервісів послужило відправною точкою для проникнення великих даних чи не в усі сфери діяльності людини. В першу чергу, науково-дослідницьку діяльність, комерційний сектор і державне управління.

Наприклад, датчики, встановлені на авіадвигуни, генерують близько 10 Тб за півгодини. Приблизно такі ж потоки характерні для бурових установок і нафтопереробних комплексів. Тільки один сервіс коротких повідомлень Twitter, незважаючи на обмеження довжини повідомлення в 140 символів, генерує потік 8 Тб/доб. Якщо всі подібні дані накопичувати для подальшої обробки, то їх сумарний обсяг буде вимірюватися десятками і сотнями петабайт. Додаткові складнощі виникають з варіативності даних: їх склад і

структура постійно змінюються при запуску нових сервісів, установці вдосконалених сенсорів або розгортанні нових маркетингових кампаній.

Компанії збирають і використовують дані самих різних типів, як структуровані, так і неструктуровані. Ось з яких джерел отримують дані учасники опитування (Cisco Connected World Technology Report):

- 74 відсотки збирають поточні дані;
- 55 відсотків збирають історичні дані;
- 48 відсотків знімають дані з моніторів і датчиків;
- 40 відсотків користуються даними в реальному часі, а потім видаляють їх. Найчастіше дані в реальному часі використовуються в Індії (62 відсотки), США (60 відсотків) і Аргентині (58 відсотків);
- 32 відсотки опитаних збирають неструктуровані дані - наприклад, відео. У цій області лідирує Китай: там неструктуровані дані збирають 56 відсотків опитаних.

Технології Big Data успішно реалізуються в різних індустріях, на інфографіці відображені основні споживачі: банки, телеком, рітейл, енергетика, медицина і управління міською інфраструктурою. Цікаво, що при всій різноманітності завдань вендорських рішення в сфері Big Data поки не набули яскраво вираженої галузевої спрямованості. Зараз ринок знаходиться на стадії активного формування.

Незважаючи на малий термін існування сектора Big Data, вже є оцінки ефективного використання цих технологій, засновані на реальних прикладах. Один з найвищих показників відноситься до енергетики - за оцінками аналітиків, аналітичні технології Big Data здатні на 99% підвищити точність розподілу потужностей генераторів. А охорону здоров'я США, завдяки Big Data, може заощадити до $ 300 млрд.

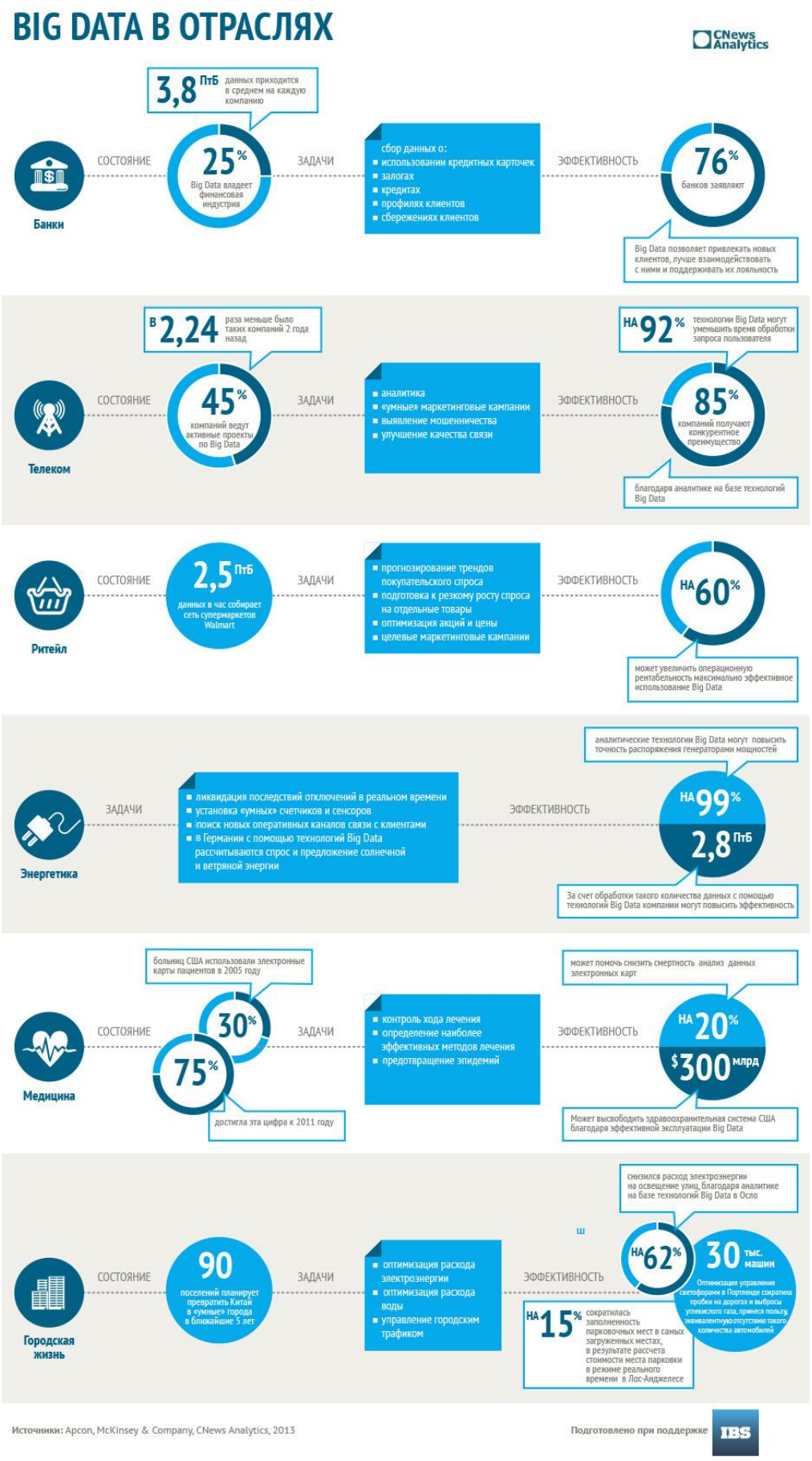


Рисунок 1.1 Big Data у різних галузях

1.2 Технології Big Data (Hadoop)

Рушійною силою реалізації BD є програмне забезпечення - як інфраструктура, так і аналітика. Основним в інфраструктурі є Hadoop. Hadoop - це велика інфраструктура програмного забезпечення для керування даними, яка використовується для розповсюдження, каталогізації, керування та запиту даних у кількох горизонтально масштабованих серверах. “Yahoo!” створений на основі відкритої вихідної інфраструктури запитів даних (випущена компанією Google) під назвою MapReduce. Він має ряд комерційно підтриманих дистрибутивів від таких компаній, як MapR Technologies та Cloudera. Hadoop є основою для обробки, зберігання та аналізу величезної кількості розподілених неструктурованих даних. Як підсистема розподіленого зберігання файлів, розподілена файлова система Hadoop (HDFS) була розроблена для обробки петабайт та екзабайт даних, що розподіляються на декількох вузлах паралельно. На рисунку 1.2 показано огляд розгортання Hadoop у великому середовищі аналізу даних.

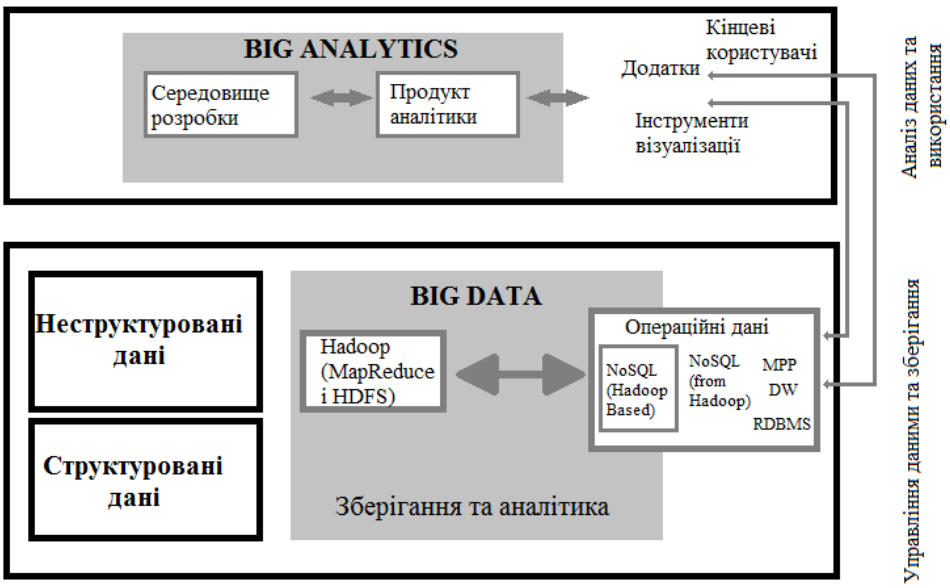


Рисунок 1.2 Структура розгортання високого рівня Hadoop

1.3Що означає BD для IT?

Завдяки поєднанню технологічних інновацій, зрілості програмного забезпечення з відкритим вихідним кодом, товаровиробничого обладнання, соціальних мереж та поширенням мобільних пристроїв, підйом великих даних створив точку перевертання, що робить збирання та аналіз даних в режимі реального часу найважливішим для бізнесу сьогодні. Однак, враховуючи, що дані та їх структури принципово різні, стає все більш очевидним, що інфраструктура, інструменти та архітектури, що підтримують аналіз даних у реальному часі, також повинні бути різними.

В якості IT-рішення великі дані відображають зростання як об’єму, так і джерел даних, а також поширеність технології в нашому повсякденному житті. Оскільки все більше і більше з того, що ми робимо, пов'язано з мережею, пристрої, які ми підключаємо до мережі, самостійно працюють на базі масиву датчиків - слід очікувати, що поточний потік даних зростатиме. У центрах обробки даних кожен вузол (сервери, накопичувачі та програми) генерує величезну кількість файлів журналів та ізольованих потоків даних, які також можна збирати, зберігати та аналізувати.

1.4Масштабована інфраструктура даних

Ще однією унікальною характеристикою великих даних є те, що, на відміну від великих наборів даних, які історично зберігалися та аналізувались, часто великі дані складаються з дискретно малих, додаткових елементів даних з додатками чи модифікаціями в режимі реального часу. Це не добре працює в традиційних сховищах обробки онлайн-транзакцій (OLTP) або в традиційних інструментах аналізу SQL. Великі дані вимагають плоскої, горизонтально масштабованої бази даних, часто із унікальними інструментами запитів, які працюють в режимі реального часу з фактичними

даними (на відміну від часових графічних знімків). У таблиці 1.1 порівнюються традиційні дані з великими даними.

Таблиця 1.1 Порівняльна таблиця BD проти традиційних типів даних

Компоненти	Традиційні дані	BD
Архітектура	Централізована	Розподілена
Об’єм	Терабайти	Від петабайт до ексабайтів
Тип даних	Структуровані або традиційні	Неструктуровані або слабоструктуровані
Зв’язки	Відомі	Комплексні/невідомі
Модель даних	Фіксована схема	Без схеми

Існують нові форми підтримки баз даних, деякі з них використовують традиційні SQL для запитів (часто називають NewSQL), а деякі з них багато в чому відмовляються від SQL на користь нових бібліотек запитів (часто називаються NoSQL). Інші компанії, які прагнуть використати свою велику інфраструктуру SQL, поставили перед собою існуючу інфраструктуру баз даних, щоб створити більш гнучке, горизонтально масштабоване середовище, щоб використовувати великі інструменти та можливості для роботи з даними. Це додало складності та створило кілька чітких рішень для IT-організацій при плануванні реалізації їх великих даних. У цьому ж середовищі вплив на високопродуктивну аналітику даних у режимі реального часу глибоко впливає на багато аспектів технологій та архітектур центрів обробки даних.

На рисунку 1.3 зображена інфраструктурна архітектура BD реалізована у хмарному середовищі.

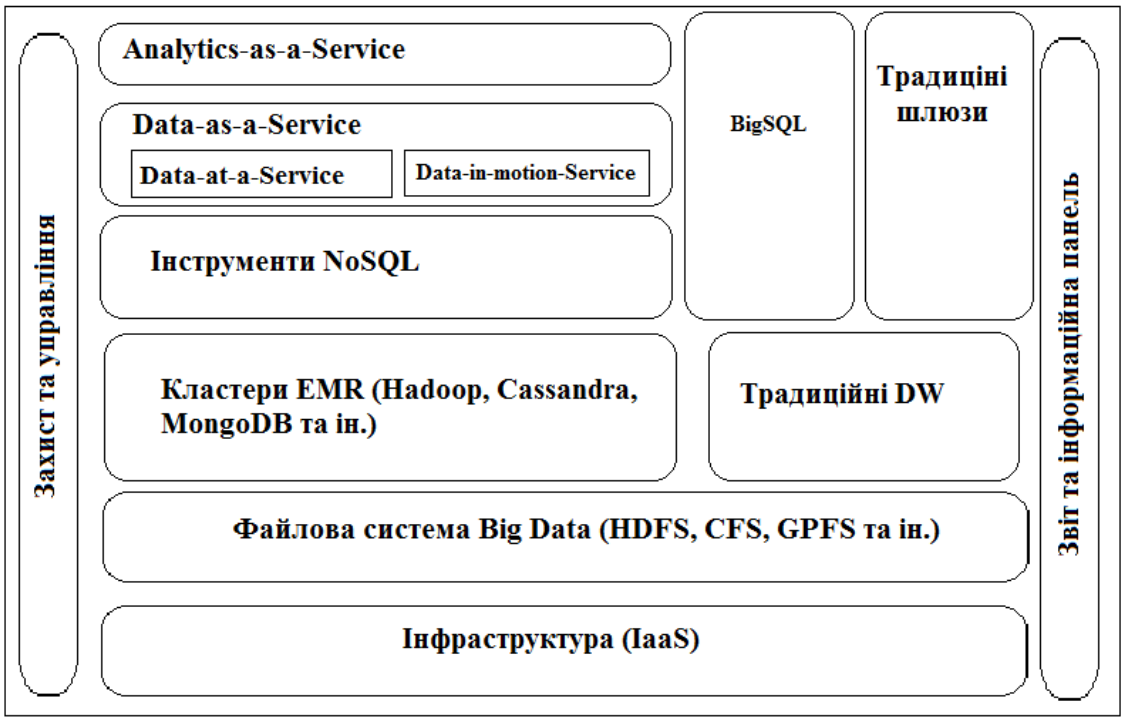


Рисунок 1.3 Інфраструктурна архітектура BD реалізована у хмарному середовищі

Інфраструктура для роботи навантажень Big Data може складатися з фізичного обладнання або може бути частиною інфраструктури хмари (IaaS). У будь-якому випадку для конкретного робочого навантаження потрібні спеціалізований пул ресурсів, наприклад, Hadoop, обробка онлайн-транзакцій (OLTP) або поточні аналітичні навантаження

Поверх інфраструктури є велика файлова система даних, яка також залежить від робочих навантажень. Тут знаходяться дані. У випадку Hadoop вибрана файлова система - це розподільна файлова система Hadoop (HDFS) (або загальна паралельна файлова система (GPFS)). HDFS працює на вершині ресурсного пулу чи кластера.

Кластер MapReduce працює на даних, що зберігаються в кластері. У випадку з Hadoop, кластер MapReduce використовує дані в HDFS або GPFS. Amazon пропонує службу хмарних служб Elastic MapReduce (EMR), яка надає функції MapReduce, але використовує сховища даних у сховищі Amazon S3. Зберігання Amazon S3 більше схоже на традиційне сховище, ніж

HDFS, де ви можете мати спеціальні кластери для різних навантажень (наприклад, для виробництва або розробки). Оскільки кластери можуть працювати на вершині служби IaaS, вам не потрібно турбуватися про втрату даних. Кластери можуть бути розширені або зменшені динамічно. Традиційні бази даних забезпечують всю необхідну функцію управління даними для збережених у них даних.

Різні NoSQL інструменти забезпечують абстрактний шар на вершині MapReduce. Інструменти, такі як Hive, можуть використовуватися для спеціальних запитів. Або Apache Pig (інструмент програмування для операцій з даними) може використовуватися для вилучення, перетворення та завантаження (ETL) та для запуску спеціальних алгоритмів проти кластера Hadoop. Hive і Pig створюють зворотні кінцеві завдання MapReduce, які виконуються на Hadoop. Тепер доступний новий набір інструментів, який надає SQL-доступ до кластерів EMR, таких як IBM BigSQL. BigSQL поєднує в собі інтерфейс SQL (для спеціальної обробки) з паралельною обробкою для обробки великої кількості даних.

Рівень Data-as-a-Service забезпечує API доступ до даних, які зберігаються в основній великій інфраструктурі даних. Цей шар можна розбити на два підмножини:

- Data-at-rest-Service: забезпечує перегляд даних, що зберігаються в нижніх шарах, за допомогою попередньо визначених перетворень даних, встановлених датавченими.
- Data-in-motion-Service: забезпечує безперервний потік перетворених та, можливо, оброблених даних для споживання.

Рівень служби Analytics-as-a-Service одержує дані Data-at-rest-Service та Data-in-motion-Service, застосовує аналітичні алгоритми, які є специфічними для будь-яких програм, що залучають дані, та включають інформаційні панелі, звіти, візуалізацію та прогнозне моделювання. пов'язані з даними. Цей шар приховує всю складність збирання, зберігання та очищення даних та забезпечує легкий механізм вивчення великих обсягів даних для дата вчених.

1.5 Методики аналізу великих даних

Існує безліч різноманітних методик аналізу масивів даних, в основі яких лежать інструменти, запозичені з статистики та інформатики (наприклад, машинне навчання). Список не претендує на повноту, проте в ньому відображені найбільш потрібні в різних галузях підходи. При цьому слід розуміти, що дослідники продовжують працювати над створенням нових методик і вдосконаленням існуючих. Крім того, деякі з перерахованих методик зовсім не обов'язково можуть бути застосовані виключно до великих даними і можуть з успіхом використовуватися для менших за обсягом масивів (наприклад, A/B-тестування, регресійний аналіз). Безумовно, чим більше об'ємний і діверсіфікований масив піддається аналізу, тим точніші і релевантні дані вдається отримати на виході.

A / B testing. Методика, в якій контрольна вибірка по черзі порівнюється з іншими. Таким чином вдається виявити оптимальну комбінацію показників для досягнення мети, наприклад, досягнення найкращою відповідної реакції споживачів на маркетингову пропозицію. Великі дані дозволяють провести величезну кількість ітерацій і таким чином отримати статистично достовірний результат.

Association rule learning. Набір методик для виявлення взаємозв'язків, тобто асоціативних правил, між змінними величинами в великих масивах даних. Використовується в data mining.

Classification. Набір методик, які дозволяє передбачити поведінку споживачів в певному сегменті ринку (прийняття рішень про покупку, відтік, обсяг споживання і ін.). Використовується в data mining.

Cluster analysis. Статистичний метод класифікації об'єктів по групах за рахунок виявлення наперед невідомих загальних ознак. Використовується в data mining.

Crowdsourcing. Методика збору даних з великої кількості джерел.

Data fusion and data integration. Набір методик, який дозволяє аналізувати коментарі користувачів соціальних мереж і зіставляти з результатами продажів в режимі реального часу.

Data mining. Набір методик, який дозволяє визначити найбільш сприйнятливі для продукту, що просувається або послуги категорії споживачів, виявити особливості найбільш успішних працівників, передбачити поведінкову модель споживачів.

Ensemble learning. У цьому методі задіюється безліч предикативних моделей за рахунок чого підвищується якість зроблених прогнозів.

Genetic algorithms. У цій методиці можливі рішення представляють у вигляді «хромосом», які можуть комбінуватися і мутувати. Як і в процесі природної еволюції, виживає найбільш пристосована особина.

Machine learning. Напрямок в інформатиці (історично за ним закріпилася назва «штучний інтелект»), яке має на меті створення алгоритмів самонавчання на основі аналізу емпіричних даних.

Natural language processing (NLP). Набір запозичених з інформатики та лінгвістики методик розпізнавання природної мови людини.

Network analysis. Набір методик аналізу зв'язків між вузлами в мережах. Стосовно до соціальних мереж дозволяє аналізувати взаємозв'язки між окремими користувачами, компаніями, спільнотами і т.п.

Optimization. Набір чисельних методів для редизайну складних систем і процесів для поліпшення одного або декількох показників. Допомагає в прийнятті стратегічних рішень, наприклад, складу виведеної на ринок продуктової лінійки, проведенні інвестиційного аналізу та ін.

Pattern recognition. Набір методик з елементами самонавчання для передбачення поведінкової моделі споживачів.

Predictive modeling. Набір методик, які дозволяють створити математичну модель наперед заданого ймовірного сценарію розвитку подій. Наприклад, аналіз бази даних CRM-системи на предмет можливих умов, які підштовхнуть абонента змінити провайдера.

Regression. Набір статистичних методів для виявлення закономірності між зміною залежної змінної і однієї або декількох незалежних. Часто застосовується для прогнозування. Використовується в data mining.

Sentiment analysis. В основі методик оцінки настроїв споживачів лежать технології розпізнавання природної мови людини. Вони дозволяють вичленувати із загального інформаційного потоку повідомлення, пов'язані з цікавлять предметом (наприклад, споживчим продуктом). Далі оцінити полярність судження (позитивне чи негативне), ступінь емоційності та ін.

Signal processing. Запозичений з радіотехніки набір методик, який має на меті розпізнавання сигналу на фоні шуму і його подальшого аналізу.

Spatial analysis. Набір частково запозичених з статистики методик аналізу просторових даних - топології місцевості, географічних координат, геометрії об'єктів. Джерелом великих даних в цьому випадку часто виступають геоінформаційні системи (ГІС).

Statistics. Наука про збір, організації та інтерпретації даних, включаючи розробку опитувальників і проведення експериментів. Статистичні методи часто застосовуються для оціночних суджень про взаємозв'язки між тими чи іншими подіями.

Supervised learning. Набір заснованих на технологіях машинного навчання методик, які дозволяють виявити функціональні взаємозв'язки в аналізованих масивах даних.

Simulation. Моделювання поведінки складних систем часто використовується для прогнозування, передбачення і опрацювання різних сценаріїв при плануванні.

Time series analysis. Набір запозичених з статистики та цифрової обробки сигналів методів аналізу повторюваних з плином часу послідовностей даних. Одні з очевидних застосувань - відстеження ринку цінних паперів або захворюваності пацієнтів.

Unsupervised learning. Набір заснованих на технологіях машинного навчання методик, які дозволяють виявити приховані функціональні

взаємозв'язки в аналізованих масивах даних. Має спільні риси з Cluster Analysis.

Visualization. Методи графічного представлення результатів аналізу великих даних у вигляді діаграм або анімації для спрощення інтерпретації полегшення розуміння отриманих результатів.

1.6 Висновки до розділу 1

Big Data - це комплексне поняття, що поєднує в собі:

1) безпосередньо дані (безліч закодованої інформації);

2) сукупність технологій роботи з цими даними;

3) новий погляд, нову парадигму в науці про дані (data science).

Були розглянуті основні положення Big Data, реалізація BD за допомогою інфраструктури та аналітики.

В основі методик аналізу масивів даних лежать інструменти, запозичені з статистики та інформатики (наприклад, машинне навчання). Крім того, деякі з перерахованих методик зовсім не обов'язково можуть бути застосовані виключно до великих даними і можуть з успіхом використовуватися для менших за обсягом масивів (наприклад, A/B-тестування, регресійний аналіз). Безумовно, чим більше об'ємний і діверсіфікований масив піддається аналізу, тим точніші і релевантні дані вдається отримати на виході.

РОЗДІЛ 2. АЛГОРИТМИ АНАЛІЗУ НЕСТРУКТУРОВАНИХ ДАНИХ

2.1 Алгоритм пошуку асоціативних правил

2.1.1 Поняття асоціативних правил

Позначення:

$I = \{i_1, i_2, \dots, i_m\}$ – множина елементів.

D – множина транзакцій, де кожна транзакція T – множина елементів, так що $T \subseteq I$. Транзакція T містить деяку множину елементів X із I якщо $X \subseteq T$. Асоціативним правилом називається імплікація $X \Rightarrow Y$, де $X \subset I$, $Y \subset I$, і $X \cap Y = \emptyset$. Іншими словами асоціативні правила це правила виду: якщо (умова), то (результат), де умова і результат представляються наборами елементів з безлічі I. Для аналізу даних в першу чергу цікаві корисні правила, тобто нетривіальні, зрозумілі правила, які відкривають нові закономірності в даних. Корисність правила визначається відповідно до наступних критеріїв:

- Підтримка (support) – показує який відсоток транзакцій підтримує дане правило. Тобто правило $X \Rightarrow Y$ має підтримку s в множині транзакцій D, якщо s% транзакцій із множини D містять $X \cup Y$.
- Достовірність (confidence) - показує ймовірність того, що з наявності в транзакції набору X слідує наявність в ній набору Y. Тобто правило $X \Rightarrow Y$ міститься у множині транзакцій D з достовірністю c, якщо c% транзакцій із множини D, які містять X також містять Y.

Завдання алгоритмів пошуку асоціативних правил - знайти всі правила, які задовольняють деяким мінімальним значенням підтримки і достовірності. Ці обмеження вводяться для того, щоб кількість знайдених правил не було занадто великим. При дуже низькому значенні підтримки алгоритм буде

генерувати величезну кількість правил, які неможливо буде обгрунтувати з точки зору статистики. Навпаки, при занадто високому значенні підтримки будуть отримані прості правила, очевидні і без проведення аналізу.

2.1.2 Алгоритм Apriori

Алгоритм Apriori для пошуку асоціативних правил використовує властивість анти-монотонності, яке полягає в тому, що підтримка будь-якого набору елементів не може перевищувати мінімальної підтримки будь-якого з його підмножин. Або, іншими словами, будь-який k-елементний набір буде часто зустрічається (тобто мати мінімальну підтримку) тоді і тільки тоді, коли всі його k-1-елементні підмножини будуть часто зустрічаються. Це властивість дозволяє алгоритму формувати нові множини, використовуючи тільки часті підмножини попереднього рівня.

Послідовність кроків алгоритму Apriori:

1. Читання вхідних даних і формування з них множини транзакцій.
2. Формування множини 1-елементних частих наборів L_1. На цьому кроці алгоритм проходить по всій базі транзакцій і рахує частоту появи кожного елемента. Якщо значення підтримки елемента перевищує мінімально заданий, то елемент додається в множину частих l-елементних наборів L_1.
3. Генерація множини кандидатів C_k. На цьому кроці формуються потенційно часто зустрічні набори елементів - такі набори називають кандидатами. k-елементні кандидати формуються шляхом об'єднання двох (k-1)-елементних часто зустрічаються наборів, що мають однакові (k-2)-елементні підмножини.
4. Підрахунок підтримки кандидатів з C_k. На цьому кроці алгоритм проходить по базі і рахує кількість входжень всіх кандидатів в усі транзакції.

5. Видалення нечастих k-елементних наборів. Для кожного k-елементного набору значення його підтримки порівнюється з мінімально заданим. Якщо значення підтримки елемента менше мінімально заданого, то об'єкт буде видалений з безлічі частих k-елементних наборів.
6. Генерація безлічі асоціативних правил. Для кожного частого набору l формуються правила: a => (l - a), де a - всі непусті підмножини множини l.

Кроки 3, 4 і 5 служать для формування множин всіх частих наборів L_k на кожній ітерації алгоритму, тому далі в роботі ці кроки будуть розглядатися як єдиний блок - формування множин всіх частих наборів L_k.

Загальна схема алгоритму приведена на рисунку 2.1.

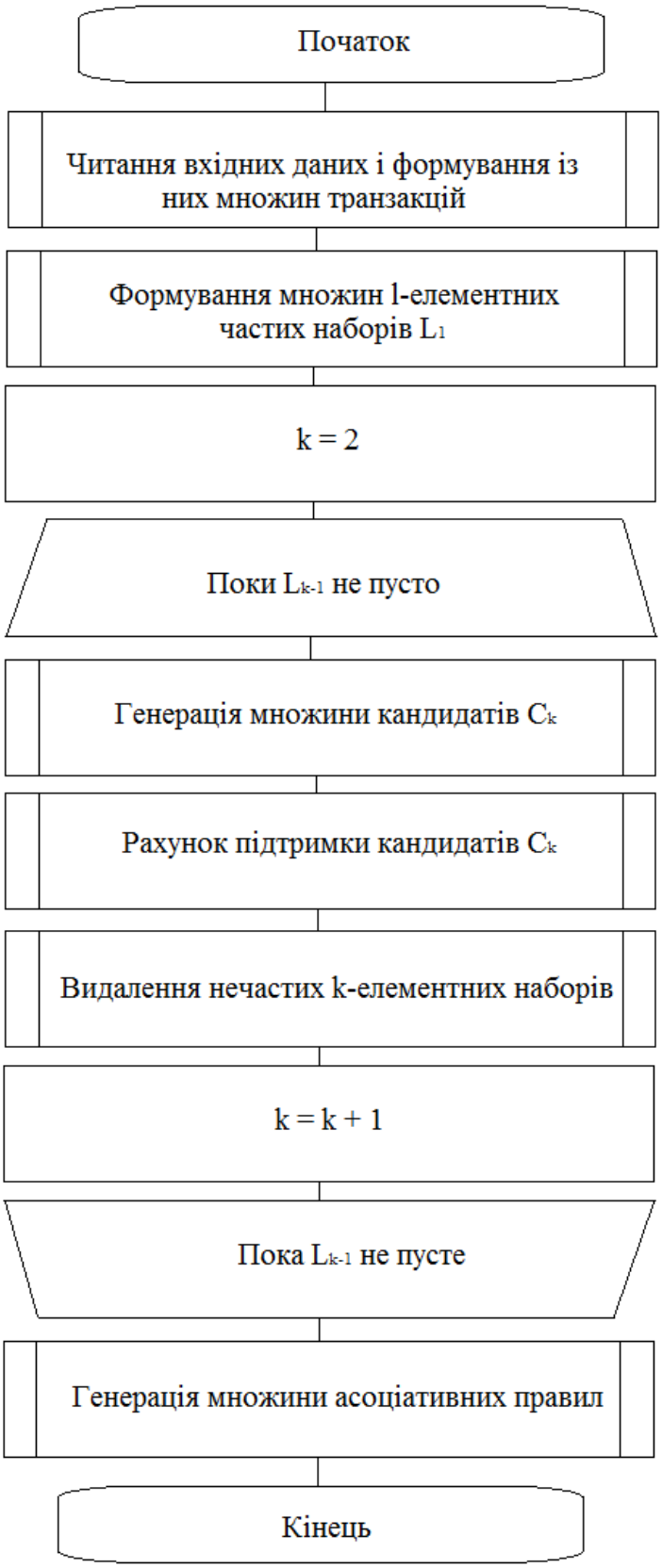
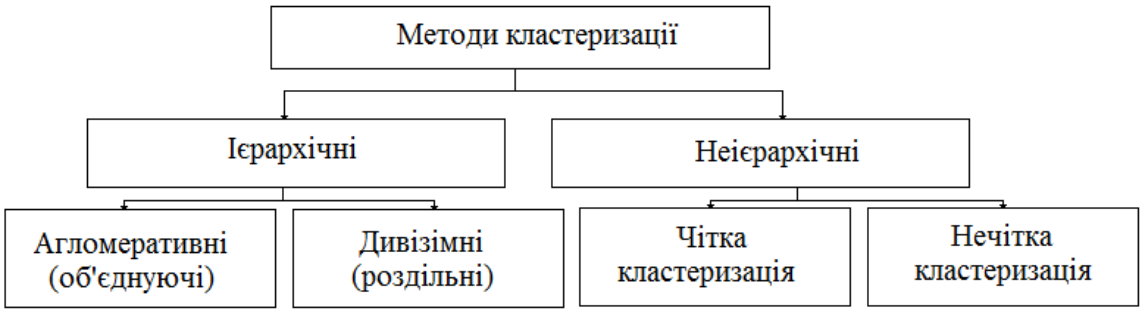


Рисунок 2.1 Загальна схема алгоритму Apriori

2.2 Алгоритми кластеризації

На рисунку 2.2 зображено класифікація кластерних методів.



Рисунки 2.2 Класифікація кластерних методів

Ієрархічна кластеризація. При ієрархічній кластеризації виконується послідовне об'єднання менших кластерів в великі або поділ великих кластерів на менші.

Агломеративні методи полягають в послідовним об'єднанні вихідних даних. Внаслідок цього відбувається зменшення числа кластерів. На початку роботи об'єднуючого алгоритму всі об'єкти сприймаються як окремі кластери і на першому кроці самі схожі об'єкти об'єднуються в кластер. На наступних кроках об'єднання триває і відбувається до тих пір, поки всі об'єкти не будуть складати один кластер. Дівізімні методи характеризується послідовним поділом вихідного кластера, що складається з усіх об'єктів, і відповідним збільшенням числа кластерів. На початку роботи алгоритму всі об'єкти належать одному кластеру, який на наступних кроках ділиться на менші кластери. Найвідомішими методами ієрархічного кластерного аналізу є метод ближнього сусіда (одиночної зв'язку), метод далекого сусіда (повної зв'язку), метод середньої зв'язку та метод Варда. Сказати точно, який з методів краще, а який гірше, не можна. Кожен має свої переваги і недоліки і підходить для визначення конкретних завдань. Наприклад, метод, одиночної зв'язку придатний для виділення кластерів так званої стрічкової форми. Метод

повного зв'язку не має такого ефекту, але на перших кроках роботи може об'єднати в кластер несхожі групи.

Ієрархічні методи дають можливість виявляти вкладені кластери, вони дозволяють побудувати дендрограмму (дерево кластерів), яка при невеликому числі об'єктів вибірки є наочним посібником для інтерпретації результатів роботи алгоритмів. У випадках обробки великих масивів даних ієрархічними методами для візуалізації результатів використовується діаграма досягнень.

Неієрархічна кластеризація. Неієрархічні методи є ітеративними методами поділу вихідної вибірки даних. У процесі такого поділу створюються нові кластери. Розподіл відбувається до того моменту, поки не буде виконано правило зупинки. Між собою неієрархічні методи розрізняються вибором початкової точки, правила формування нових кластерів і правилом зупинки. Найчастіше на практиці застосовується алгоритм к-середніх. Він має на увазі, що аналітик заздалегідь фіксує кількість кластерів в результуючому розбитті. При цьому кластери повинні бути максимально різні. Вибір числа кластерів повинен грунтуватися на результатах попередніх досліджень, теоретичних міркуваннях або інтуїції аналітика або експерта. Однак, існує думка, що підхід, при якому число кластерів визначається аналітиком заздалегідь, не можна вважати гнучким.

Чіткі (або непересічні) алгоритми кожному об'єкту вибірки ставлять у відповідність номер кластера, тобто кожен об'єкт належить тільки одному кластеру.

Нечіткі (або пересічні) алгоритми кожному об'єкту ставлять у відповідність набір речових значень, що показують ступінь відносини об'єкта до кластерів. Таким чином, кожен об'єкт відноситься до кожного кластеру з певною ймовірністю. Алгоритм к-середніх фактично є прообразом практично всіх алгоритмів нечіткої кластеризації.

Порівнюючи ієрархічні і неієрархічні методи, автор статті [4] звертає увагу на наступні аспекти. Неієрархічні методи виявляють більш високу

стійкість по відношенню до викидів, невірного вибору метрики, включенню незначущих змінних в базу для кластеризації та ін. Але при цьому дослідник повинен заздалегідь фіксувати результуючу кількість кластерів, правило зупинки і, якщо на те є підстави, початковий центр кластера. Останній момент істотно відбивається на ефективності роботи алгоритму. Якщо немає підстав штучно задати цю умова, рекомендується використовувати ієрархічні методи. Потрібно врахувати також ще один момент, істотний для обох груп алгоритмів: не завжди правильним рішенням є кластеризація всіх спостережень. Можливо, більш ефективним буде спочатку очистити вибірку від аномальних даних, а потім продовжити аналіз. Можна також не ставити дуже високий критерій зупинки (можна робити зупинку, наприклад, коли кластеризовано більше 80% спостережень).

На практиці найкраще зарекомендували себе гібридні підходи, де шліфування кластерів виконується методом к-середніх, а початкове розбиття - одним із сильніших методів [5].

Автори статті [6] вказують, що ієрархічний аналіз є розвідувальним шляхом для послідовного кластерного аналізу. Він дозволяє виділити кількість кластерів, склад й властивості яких можна отримати методом k-середніх.

В кінці дев'яностих початку двохтисячних років було розроблено безліч алгоритмів, в яких методи ієрархічної кластеризації інтегровані з іншими методами: BIRCH (США, 1996 г.), CURE (США, 1998 г.), OPTICS (Німеччина, 1999 г.). На базі неієрархічних методів були створені алгоритми DBSCAN, OPTICS (Німеччина, 1996. / 1999 г.), MAFIA (США, 1999 г.), STING (США, 1997), CLIQUE (США, 1998 г.), BRIDGE (Сінгапур , 2001 г.).

Ієрархічний алгоритм BIRCH використовує двоступеневу кластеризацию, застосовується для великих масивів даних, працює тільки числами, масштабується, використовує обмежений обсяг пам'яті. Недоліки роботи алгоритму: визначає тільки сферичні кластери, необхідне задання порогових значень щільності.

DBSCAN відноситься до неієрархічних обчислювально ефективних алгоритмів. Для його роботи потрібно завдання параметрів досяжності і зв'язності. В даний час існує багато модифікацій DBSCAN, одна з яких - алгоритм OPTICS.

Неієрархічні алгоритм BRIDGE є трьохетапним комбінованим. Згідно його роботі вибірка спочатку обробляється алгоритмом к-середніх (або BIRCH), потім визначення викидів використовується DBSCAN. На третьому етапі знову застосовується к-середніх до вибірки очищеної від «шуму». Застосування такої комбінації алгоритмів дозволяє виключити недоліки обох методів.

Обчислювальна складність алгоритмів (O) дозволяє визначити, як буде збільшуватися час на виконання алгоритму. Інформація по обчислювальній складності наведена в таблиці 2.1.

Таблиця 2.1 Обчислювальна складність алгоритмів кластеризації

Алгоритм кластеризації	Обчислювальна складність
Ієрархічні	O(n^2)
k-середніх	O(nkl),
c-середніх	де k – число кластерів, l – число ітерацій
Виділення зв’язних компонент	Залежить від алгоритмів
Мінімальне покриття дерева	O($n^2 \log n$)
Пошарова кластеризація	O($n^2 \max(n, m)$), де $m < \frac{n(n-1)}{2}$

Вважається, що кластерний аналіз пред'являє наступні вимоги до вихідних даних:

1. Показники не повинні корелювати між собою.

2. Показники повинні бути безрозмірними.

3. Розподіл показників повинно бути близько до нормального.

4. Показники повинні відповідати вимогам «стійкості», під якою розуміється відсутність впливу на їх значення випадкових факторів.

5. Вибірка повинна бути однорідна, не містити «викидів».

Ці вимоги породжують виникнення деяких проблем класичного кластерного аналізу, пов'язаних з тим, що [7]:

а) параметри узагальненого образу кластера обчислюються як середні від вихідних об'єктів (класів) або центри тяжкості;

б) як критерій подібності використовується евклідова відстань або його варіанти, некоректні в неортонормованих просторах, які зазвичай і зустрічаються на практиці;

в) кластерний аналіз проводиться на основі вихідних змінних або матриці спряженості, що залежать від одиниць вимірювання по осях, для формалізації яких використовуються шкали різних типів.

З цими проблемами пов'язане те, що результати кластеризації часто незрозумілі фахівцям і не піддаються змістовній інтерпретації, не узгоджуються з оцінками експертів, їх досвідом і інтуїтивними очікуваннями[7].

2.3 Алгоритми задач класифікації та регресії

Дерева регресії і класифікації, відомі також під загальною назвою як дерева рішень (Decision Tree - DT), являють собою структури даних, що дозволяють інтерпретувати шаблони даних з метою їх розпізнавання. Дерева рішень організовані у вигляді ієрархічної структури, що складається з вузлів прийняття рішень по оцінці значень певних змінних для прогнозування результуючого значення. Застосування дерев класифікації призводить до отримання символічного позначення класу, а в результаті використання дерев регресії відбувається повернення безперервних значень.

Перш ніж приступати до розгляду того, що дозволяють досягти дерева рішень, або навіть до розгляду питання про їх побудові, необхідно зрозуміти основні концепції, що лежать в основі такої структури даних, як дерево рішень.

Будь-яке дерево рішень виводить прогнозоване значення, отримане в результаті оцінки деяких вхідних атрибутів. Дерева рішень поділяються на два різних типи: дерева класифікації і дерева регресії. Ця різниця не залежить від типів вхідних даних, оскільки дерева того і іншого типів можуть приймати або безперервні, або символічні значення. Визначальним фактором, від якого залежить тип дерева, є вихідне значення. Дерево рішень з безперервними вихідними значеннями іменується деревом регресії, а дерева класифікації замість цього виводять конкретні значення. Підсумкові відомості, що визначають відмінності між типами дерев рішень, наведені в таблиці 2.2.

Таблиця 2.2 Різниця між двома типами дерев рішень, які визначаються тим, результат якого типа вони повертають

Тип дерева	Прогнозування	Тип даних
Дерево класифікації	Дискретне	Символи
Дерево регресії	Неперервне	Дійсні числа

Вибірки даних призначені для обробки деревами рішень і часто служать також в якості визначення формату вводу-виводу.

Будь-яка вибірка даних являє собою безліч атрибутів, які прийнято також називати змінними прогнозування. Кожен атрибут може являти собою безперервне значення (тобто число з плаваючою точкою) або символ (тобто безліч неврегульованих дискретних значень). Такі атрибути дозволяють концептуально уявити майже будь-яку інформацію.

Передбачена також можливість застосовувати додаткові атрибути, що мають спеціальне значення, які відомі під назвою змінних відгуку (або

залежних змінних). Такий атрибут може бути виражений за допомогою символу, що представляє дискретні категорії (в деревах класифікації) або безперервне значення (в деревах регресії). Змінні відгуку можуть служити в якості критеріїв для прийняття рішень по відношенню до кожної з вибірок при вирішенні задач обох типів - і класифікації, і регресії.

Будь-яке дерево рішень по суті являє собою деревовидний граф, в буквальному сенсі цього поняття, сформульованого в комп'ютерних науках. Ця структура даних складається з вузлів, з'єднаних один з одним ребрами (рис. 2.3). При цьому не допускається, щоб ребра утворювали цикл, так як в противному випадку дерево перетворюється в граф, відмінний від деревовидного (а при використанні такого графа для прийняття рішень виникають труднощі).

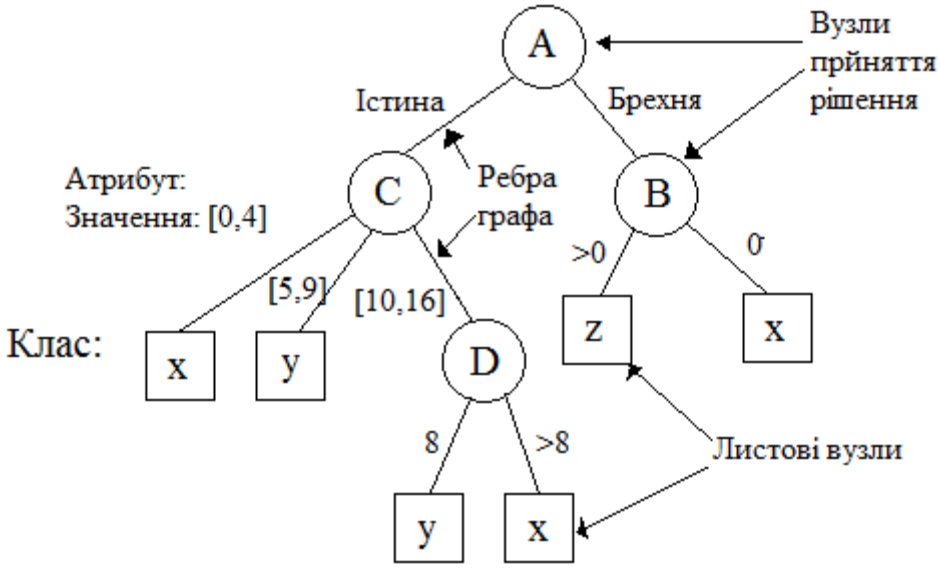


Рисунок 2.3 Просте дерево рішень

У дереві є один особливий вузол, відомий як кореневий. По суті, цей вузол є основою дерева, оскільки від кореня можна перейти по дереву до будь-якого іншого вузла. Ще до однієї особливого різновиду вузлів відносяться вузли, що знаходяться в кінці будь-якої ланцюжка поспіль ребер, - листові вузли.

Кожен рівень в дереві може розглядатися як одне з рішень; вузол прийняття рішень забезпечує перевірку умови, а кожне ребро позначає один з можливих варіантів. Більш формально можна відзначити, що вузли прийняття рішень містять критерії вибору, а ребра висловлюють взаємовиключні результати перевірки відповідності цим критеріям. По суті, при кожній перевірці умови відбувається сортування вибірок даних таким чином, що кожен елемент даних визначається як відповідний тільки одному ребру. Якщо всі вибірки розглядаються як одне загальне безліч даних, то критерії прийняття рішень розбивають цю множину на непересічні підмножини. В результаті об'єднання таких перевірок в деяку ієрархію фактично організовується процес розбиття всіх даних на все менші частини, що відбувається до тих пір, поки не досягається листовий вузол. Кожен листовий вузол відповідає невеликій, але виключній (неповторюваній) частині вихідної множини.

Оскільки атрибути можуть виражатися у вигляді символів або безперервних значень, самі перевірки можуть бути організовані у вигляді булевих умов або безперервних відносин. А від кількості можливих результатів перевірки залежить те, скільки ребер має виходити з розглянутого вузла прийняття рішень. Нижче перераховані найбільш часто використовувані перевірки умов.

- *Перевірка мулевого значення.* При проведені такої перевірки визначається, чи призводить використання якого-то конкретного оператора до отримання істинного чи помилкового значення. Можливими результатами перевірки буде істина чи брехня.
- *Визначення знаку.* При виконанні такої перевірки визначається знак вираження. Результатом може були або позитивне, або від’ємне значення. Вказана перевірка може розглядатися як окремий випадок перевірки булевого значення.
- *Перевірка приналежності до класу.* При виконанні такої перевірки визначається до якого класу відноситься даних символ.

Результатом перевіки буде позначення одиного із можливих класів.

- *Перевірка приналежності до області значень.* При проведені такої перевірки повинно бути з’ясовано, до якої області значень відноситься дане значення. Кожний із можливих результатів вказує, до якої області значень, на які ділиться вся область значень змінних, відноситься дане значення. Така перевірка може розглядатися як перевірка приналежності до класу.

Після створення дерева рішень з'являється можливість оцінювати значення змінної відгуку для невідомих вибірок даних з урахуванням лише значень атрибутів. При виконанні такої операції одна вибірка обробляється за одною, що дозволяє перевіряти на відповідність критеріям і оцінювати цілі набори даних.

Обробка кожної вибірки даних і, зокрема, визначення класу і (або) значення для вибірки здійснюються шляхом переходу по дереву, починаючи від кореневого вузла, в напрямку до листових вузлів. Кожен етап переходу здійснюється з урахуванням результатів перевірки умов, а вибір напрямку переходу відбувається з урахуванням змінних прогнозування (рис. 2.4).

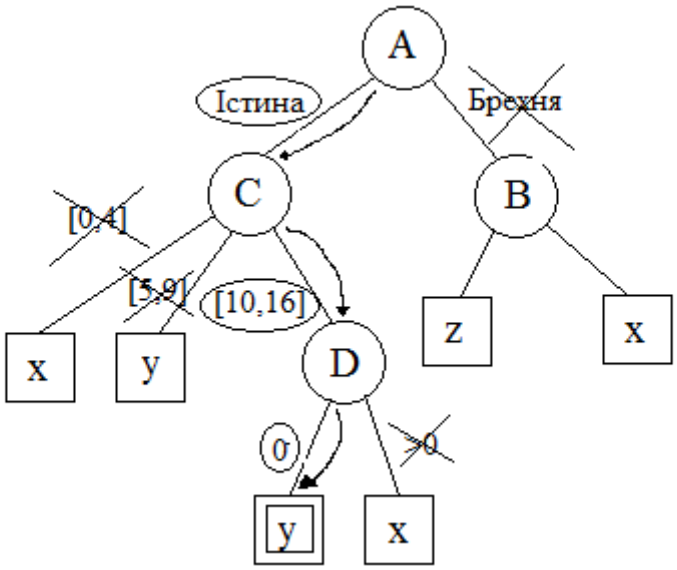


Рисунок 2.4 Використання вибірки даних для управління переходом по дереву рішень

На кожному рівні дерева рішення приймається на основі значень атрибутів. Для оцінки значень змінних прогнозування використовуються перевірки умов в кожному вузлі. Результат оцінки завжди відповідає тільки одному з ребер, що виходять з вузла прийняття рішень. Наявність в будь-яких обставинах тільки одного застосовного ребра гарантовано, оскільки умови є повними (тобто допускають не менш одного варіанта рішення) і взаємовиключними (тобто допускають тільки один варіант рішення). Під час переходу простежується ребро, що веде до наступного вузла прийняття рішень, в якому цей процес повторюється.

Перехід по дереву припиняється після досягнення кінця ланцюжка ребер. Кожен листовий вузол в дереві відповідає одному з класів (або певному значенню), тому вибірка даних, яка використовувалася для проходження по дереву, повинна належати даному класу (або мати значення). Дерево рішень гарантує, що кожна вибірка буде відповідати одному і тільки одному листовому вузлу, і тому їй буде присвоєно єдине позначення класу або одна оцінка.

З концептуальної точки зору, процес застосування дерева рішень є надзвичайно простим, і це дозволяє зрозуміти, з чим пов'язана його ефективність. Найбільша складність полягає в проектуванні програмного забезпечення, що дозволяє реалізувати якийсь гнучкий спосіб моделювання дерев рішень.

2.4 Генетичні алгоритми

Генетичні алгоритми (ГА) базуються на теоретичних здобутках синтетичної теорії еволюції, яка враховує мікробіологічні механізми успадкування ознак в природних і штучних популяціях організмів, а також на накопиченому людством досвіді в селекції тварин і рослин.

Методологічна основа ГА грунтується на гіпотезі селекції, яка в узагальненому вигляді може бути сформульована так: чим вище

пристосованість особини, тим вище ймовірність того, що в потомстві, отриманому з її участю, ознаки, що визначають пристосованість, будуть виражені ще сильніше. Оскільки ГА мають справу з популяціями постійної чисельності, особливої актуальності тут нарівні з відбором в батьки набуває відбір на елімінування. Стратегія елімінування, покликана відповісти на питання «Від яких особин ми можемо безболісно відмовитися?» Становить не менш важливу компоненту сучасних ГА, ніж стратегія відбору в батьківську групу. Найчастіше особи, що володіють низькою пристосованістю, не тільки не беруть участь в генерації нового покоління, а елімінуються з популяції на поточному дискретному кроці (епохи) еволюції.

Втім, сказане справедливо не тільки для ГА, а для будь-якого чисельного методу оптимізації. Сама ідея оптимальності, як вірно підмічено в (посилання), прийшла в науку з біології. Однак далеко не завжди ми віддаємо собі звіт в тому, наскільки багато методичні прийоми оптимального проектування мають коріння в селекційній практиці і є прикладом нашого не завжди усвідомленого наслідування Природі.

Зазвичай проектування починають з формування в пошуковому просторі області допустимих значень змінних і вибору в ній деяких пробних точок.

Далі ітеративно виконують такі дії. Спочатку за допомогою математичної моделі пристрою проводять відображення точок з пошукового простору на простір критеріїв, що дозволяє зіставити уявлення про рельєф поверхні критеріїв. Потім на підставі отриманої інформації і відповідно до обраної пошукової стратегією здійснюють деякі маніпуляції з координатами точок в просторі змінних, що завершуються генерацією координат нових пробних точок.

Спосіб, який практикує опис технічних об'єктів за допомогою векторів змінних проектування має на увазі символьне кодування інформації про об'єкт. Вектор змінних - навіть не креслення, тобто дивлячись на нього і не знаючи правил кодування, неможливо скласти уявлення про об'єкт. У

певному сенсі можна стверджувати, що категорія "вектор змінних проектування" грає в техніці ту ж роль, що і категорія "генотип" в біології. Групуючи ключові параметри об'єкта в вектор змінних, ми, по суті, надаємо їм статус генетичної інформації. Саме генетичної, тому що, з одного боку, її досить, для того, щоб побудувати сам об'єкт (гіпотетично - виростити його), а по-друге, вона служить вихідним матеріалом при генерації генотипів об'єктів наступного покоління.

А адже саме такий зміст - генотипів нащадків - мають координати згадуваних нових пробних точок. Подібно до того, як в Природі схрещування організмів здійснюється на генетичному рівні, в процедурі оптимізації координати нових пробних точок виходять як результат маніпулювання координатами старих. Причому, і тут незримо присутня гіпотеза селекції - в якості батьківських завжди виступають кращі в фенотипічному відношенні, а не довільні точки (особини) з популяції потенційних рішень, невдалі ж рішення відкидають на поточному кроці (можна вважати, що вони вимирають).

Тут ми підходимо, нарешті, до того, що саме відрізняє ГА на тлі інших численних методів оптимізації.

ГА запозичують з біології:

• понятійний апарат;

• ідею колективного пошуку екстремуму за допомогою популяції особин;

• способи подання генетичної інформації;

• способи передачі генетичної інформації в низці поколінь (генетичні оператори);

• ідею про переважне розмноженні найбільш пристосованих особин (мова йде не про те, чи дасть ця особина нащадків, а про те, скільки буде у неї нащадків).

2.3.1 Представлення генетичної інформації

Подібно до того, як природний хромосомний матеріал являє собою лінійну послідовність різних комбінацій чотирьох нуклеотидів (А - аденін, Ц - цитозин, Т - тимін і Г - гуанін), вектора змінних в ГА також записують у вигляді ланцюжків символів, використовуючи двох-, трьох- або чотирибуквений алфавіт. Для простоти викладу розглянемо випадок бінарного кодування, що використовується при моделюванні еволюції гаплоїдних популяцій.

Отже, будемо вважати, що кожна змінна x_i кодується певним фрагментом хромосоми, що складається з фіксованої кількості генів (див. рис. 2.5). Все локуси хромосом діаллельни - тобто в будь-якій позиції фрагмента може стояти як нуль, так і одиниця. Поруч стоять фрагменти не відокремлюють один від одного якими-небудь маркерами, тим не менш, при декодуванні хромосоми в вектор змінних протягом усього моделируемого періоду еволюції використовується одна і та ж маска картування.

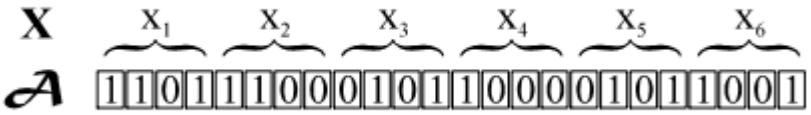


Рисунок 2.5 Найпростіша маска картування хромосоми, що визначає план розподілу спадкової інформації по довжині хромосоми

Хоча ми постійно говоримо про декодування, насправді, пряма операція, що розуміється як операція кодування вектора змінних x в хромосому A, в ГА не застосовується. Хромосоми генеруються випадковим чином, шляхом послідовного заповнення розрядів (генів), відразу в бінарному вигляді, і всякі подальшій зміні в популяції зачіпають спочатку генетичний рівень, а тільки потім аналізуються фенотипні наслідки цих змін, але ніколи не навпаки.

В принципі, для декодування генетичної інформації з бінарної форми до десяткового виду підходить будь-який двійковий-десятковий код, але зазвичай виходять з того, що вона представлена в коді Грея. Таблиця 2.3 відтворює в повному обсязі процедуру декодування фрагмента хромосоми в проекцію вектора змінних x_i.

Від коду Грея переходимо до двійково-десяткового коду, а від нього - до натуральних цілих числем. Відношення отриманого числа до максимального числа, доступному для кодування даними кількістю розрядів фрагмента (за таблицею 2.3 - число 15) і дає шукане значення зсуву змінної щодо лівої межі a_i допустимого діапазону її зміни, нормованого на ширину $b_i - a_i$ діапазону.

Таблиця 2.3 Декодування фрагментів хромосом в проекції вектора змінних

Код Грея	Двоїчно-десятичнийкод	Десятичне значення зсуву	Дійсні значення координат
0000	0000	0	a_i
0001	0001	1	$a_i + 1(b_i - a_i)/15$
0011	0010	2	$a_i + 2(b_i - a_i)/15$
0010	0011	3	$a_i + 3(b_i - a_i)/15$
0110	0100	4	$a_i + 4(b_i - a_i)/15$
0111	0101	5	$a_i + 5(b_i - a_i)/15$
0101	0110	6	$a_i + 6(b_i - a_i)/15$
0100	0111	7	$a_i + 7(b_i - a_i)/15$
1100	1000	8	$a_i + 8(b_i - a_i)/15$

Продовження таблиці 2.3

1101	1001	9	$a_i + 9(b_i - a_i)/15$
1111	1010	10	$a_i + 10(b_i - a_i)/15$
1110	1011	11	$a_i + 11(b_i - a_i)/15$
1010	1100	12	$a_i + 12(b_i - a_i)/15$
1011	1101	13	$a_i + 13(b_i - a_i)/15$
1001	1110	14	$a_i + 14(b_i - a_i)/15$
1000	1111	15	b_i

З таблиці добре видно, чому код Грея має явні переваги в порівнянні з двійково-десятковим кодом, який за певного збігу обставин породжує своєрідні тупики для пошукового процесу. Як приклад розглянемо будь-які три поруч стоять рядки з таблиці 2.3, наприклад, що кодують зрушення в 4, 5 і 6 одиниць.

Припустимо, фрагменти хромосом, що стоять в п'ятому рядку і кодують число 5, належать оптимальному вектору, що є рішенням деякої задачі, а найкраща особина з поточної популяції містить фрагмент хромосоми з рядка 4. Така ситуація сприятлива для обох кодів. Досить виконати всього одну операцію - замінити в четвертому розряді фрагмента 0 на 1 - і рішення буде знайдено. Більш цікавий випадок виходить, якщо найкраща особина містить фрагмент з рядка 6. Для коду Грея ця ситуація нітрохи не складніше попередньої - заміна 0 на 1 в третьому розряді знову призведе до успіху. У той же час двійково-десятковий код ставить нас в необхідність виконати послідовно дві операції - замінити 1 на 0 в третьому розряді і 0 на 1 в четвертому. З якою б з них ми не почали, результат не наблизить нас до вирішення (перший варіант заміни перемістить нас в четвертий рядок, а другий - взагалі в сьому). Але ж це ще не найгірший

приклад - працювати з поєднаннями 3-4, 7-8, 11-12 і т. д. рядків в двійково-десятковому коді ще складніше. Інакше кажучи, якщо залучити геометричні інтерпретації, код Грея гарантує, що дві сусідні, що належать одному ребру, вершини гіперкуба $\propto^L$, на якому здійснюється пошук, завжди декодируются в дві найближчі точки простору дійсних чисел $\Re^N$, віддалені один від одного на одну дискретну точність . Двійково-десятковий код подібною властивістю не володіє.

2.3.2 Генетичні оператори

Ті механізми передачі спадковості, які діють в Природі, і спрощена форма яких покладена в основу того, що ми називаємо генетичними операторами, насправді, слід розглядати як переможців, які здобули верх в напруженій багатовіковій боротьбі над конкурентами і відшліфованих природним відбором в такій же мірі , як і всі, що нас оточують. Сьогодні зрозуміло, що генетичні оператори могли бути запозичені не тільки з мікробіологічних досліджень, а й з аналізу мовних явищ (досить проаналізувати комбінаторні евристики, що застосовуються людиною при вирішенні кросвордів) або винахідницької діяльності (сильця). Але це сьогодні; а двадцять років тому потрібно було володіти геніальністю Дж. Холланда, щоб здогадатися, як інтерпретувати принципи дії "біологічних" механізмів для вирішення завдань адаптації в штучних системах.

Чи не головним підсумком майже чвертьстолітнього періоду дослідження самих ГА стало розуміння прекрасної взаємної компліментарності тріади генетичних операторів «кросовер - мутація - інверсія». Впливаючи з певною ймовірністю на генотипи батьківських особин, кожен з них, з одного боку, забезпечує передачу потомству життєво важливих ознак, а з іншого - підтримує протягом еволюційно значимого періоду досить високий рівень його мінливості. Відщіплення в потомстві нових, відмінних від батьківських, фенотипічних ознак відкриває для

популяції додаткові можливості для адаптації, тобто сприяє збереженню нею пошукової здатності.

Отже, оператор мутації (див. рис. 2.6), подібно до точкових мутацій в Природі, інтерпретується як заміна існуючого аллельного стану окремого гена в хромосомі на протилежне (одиниці - на нуль і навпаки). Очевидно, що в залежності від того, в якому розряді фрагмента, що кодує змінну, відбудеться мутація, залежить величина відстані, що відокремлює нащадка від батьків (мова йде не про хеммінговий простор $\propto^L$, де ця відстань дорівнює 1, а про простір дійсних чисел $\Re^N$). Інверсія призводить до порушення порядку проходження фрагментів хромосом у нащадка в порівнянні з батьківською хромосомою. Нарешті, кросовер, що описує механізм гаметогенезу в диплоїдних популяціях організмів і привнесений Холландом в моделювання еволюції гаплоїдних популяцій, призводить до того, що хромосома нащадка включає два фрагмента, один з яких належав раніше, умовно кажучи, батьківській хромосомі, а інший – материнській. Саме завдяки наявності кроссоверних обмінів особини популяції обмінюються між собою генетичною інформацією, тобто пошук набуває дійсно колективний характер.

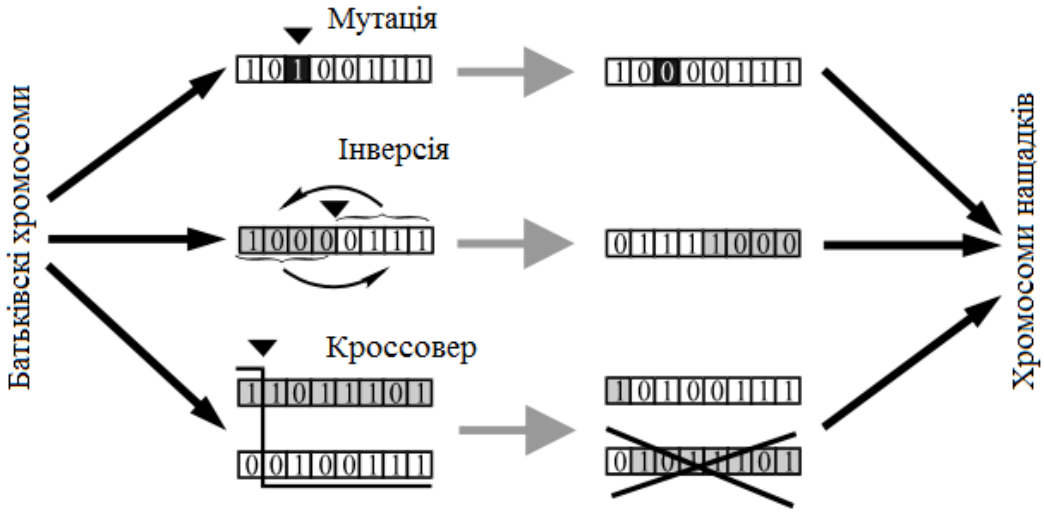


Рисунок 2.6 Тріада генетичних операторів

Іноді, кажучи про тріаду генетичних операторів, підкреслюють здатність кросовера і інверсії до глобального пошуку, в той час як мутацію ототожнюють із засобами локального налаштування рішення, відводячи їй фонову роль. Такий розподіл ролей представляється суперечним, так як мутація може породити нащадка далеко за межами локального екстремуму, в якій знаходиться батько, з іншого боку, кросовер, проведений над гаметами батьків, розташованих в загальному екстремумі, напевно породить нащадків в цьому ж екстремумі. Важливо інше - ні кросовер, ні мутація не спираються в процесі генерування нащадка на знання локального рельєфу поверхні цільової функції. У цьому сенсі їх можна вважати глобальними.

2.4 Висновки до розділу 2

В даному розділі ми розглянули 4 алгоритми, а саме алгоритми пошуку асоціативних правил, алгоритми кластеризації (метод агломерації та ділення), алгоритми задач класифікації та регресіїї та генетичні алгоритми.

Кожний алгоритм існує і працює в повсягденому житті і має як свої мінуси так і плюси. У наступному розділі ми будемо досліджувати ці алгоритми і виберемо, той алгоритм який найкраще себе підходить для аналізу неструктурованих та слабо структурованих даних.

РОЗДІЛ 3. ДОСЛІДЖЕННЯ АЛГОРИТМІВ АНАЛІЗУ НЕСТРУКТУРОВАНОЇ ТА СЛАБО СТРУКТУРОВАНОЇ ІНФОРМАЦІЇ

Неефективне управління інформацією веде до збільшення ризиків для різних форм бізнесу: зберігання персональних даних та іншої конфіденційної інформації на загальнодоступних інформаційних ресурсах, поява підозрілих призначених для користувача зашифрованих архівів, порушення політик доступу до важливої інформації і т.д.

За цих обставин вміння якісно аналізувати інформацію і оперативно реагувати на будь-які невідповідності її зберігання політикам і вимогам бізнесу є ключовим показником зрілості інформаційної стратегії організації.

Слабоструктурована інформація (CCI) - це дані, для яких визначені деякі правила і формати, але в найзагальнішому вигляді. Наприклад, рядок з адресою, рядок в прайс-листі, ПІБ і т. п. На відміну від неструктурованих, такі дані з меншими зусиллями перетворюються до структурованої формі, однак без процедури перетворення вони теж непридатні для аналізу.

До неструктурованою інформації (HI) відносяться дані, довільні за формою, що включають тексти і графіку, мультимедіа (відео, мова, аудіо). Ця форма представлення даних широко використовується, наприклад, в Інтернеті, а самі дані надаються користувачеві у вигляді відгуку пошуковими системами.

Проведемо порівняння алгоритмів аналізу неструктурованої і слабоструктурованої інформації. До таких алгоритмів відносяться наступні:

- алгоритми пошуку асоціативних правил;
- алгоритми кластеризації (методом агломерації, ділення);
- алгоритми задач класифікації і регресії (в т.ч. прогнозування часових рядів);
- алгоритми побудови нейромереж і генетичні алгоритми.

В якості критеріїв для порівняльного аналізу виберемо наступні:

1. A_1 – точність розбиття на групи;
2. A_2 – точність прогнозу;
3. A_3 – точність знаходження закономірностей;
4. A_4 – аналіз погано формалізованих даних;
5. A_5 – знаходження прихованих закономірностей даних.

Для визначення ваги критеріїв скористаємося аналітичної ієрархічної процедури Сааті [8].

Обчислення вагових коефіцієнтів виконується в наступному порядку:

1. Експертом заповнюється матриця парних порівнянь розміром $n \times n$, де n – кількість критеріїв якості. Правила заповнення матриці парних порівнянь представлені в таблиці 3.1.

Таблиця 3.1 Значення коефіцієнтів матриці парних порівнянь

X_{ij}	Значення
1	i-ий і j-тий критерій приблизно рівноцінні
3	i-ий критерій трохи переважає j-тий
5	i-ий критерій переважає j-тий
7	i-ий критерій значно переважає j-тий
9	i-ий критерій явно переважає j-тий

Якщо i-ий критерій слабший, ніж j-тий, то вказуються обратні оцінки ($\frac{1}{3}, \frac{1}{5}, \frac{1}{7}, \frac{1}{9}$). Можуть використовуватися проміжні оцінки (2, 4, 6, 8 і $\frac{1}{2}, \frac{1}{4}, \frac{1}{6}, \frac{1}{8}$). На головній діагоналі ставляться одиниці.

Матриця парних порівнянь представлена у таблиці 3.2.

Таблиця 3.2 Матриця парних порівнянь.

	A_1	A_2	A_3	A_4	A_5
A_1	1	$\frac{1}{3}$	3	7	5
A_2	3	1	5	9	7
A_3	$\frac{1}{3}$	$\frac{1}{5}$	1	5	7
A_4	$\frac{1}{7}$	$\frac{1}{9}$	$\frac{1}{5}$	1	5
A_5	$\frac{1}{5}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{5}$	1

2. Обчислюємо оцінки якості критерії – середній геометричний строк матриці парних порівнянь:

$$k_i = \sqrt[n]{\prod_{j=1}^{n} x_{ij}} \tag{3.1}$$

де n – кількість якості критеріїв.

Алгоритм обчислення середнього геометричного складається із наступних кроків:

1) перемножуємо елементи кожного рядка і записуємо отримані результати в стовпець;
2) добуваємо корінь n-ної степені із кожного елемента знайденого стовпця;
3) додаємо елементи цього стовпця;
4) ділимо кожний із цих елементів на отриману суму.

Нормалізовану оцінку для i-тої оцінки якості розраховуємо за наступною формулою:

$$\acute{\mathrm{k}}_i = \frac{k_i}{\sum_{j=1}^{n} k_j} \tag{3.2}$$

де i – позначення критерію по строчці матриці парних порівнянь. Користуючись способом приблизного обчислення власних елементів матриці порних порівнянь, обислюємо власний стовбець (вектор пріоритетів) для розглянутих критеріїв.

Нормалізовані оцінки вектора пріоритетів і є вагою критеріїв. Розглянутий підхід відповідає процесу встановлення відносної важливості об’єктів по методу Т.Сааті.

Таблиця 3.3 Матриця парних порівнянь, середні геометричні та вага критеріїв

	A_1	A_2	A_3	A_4	A_5	Середнє геометричне	Вага критеріїв
A_1	1	$\frac{1}{3}$	3	7	5	0,93	0,14
A_2	3	1	5	9	7	3,94	0,40
A_3	$\frac{1}{3}$	$\frac{1}{5}$	1	5	7	1,18	0,18
A_4	$\frac{1}{7}$	$\frac{1}{9}$	$\frac{1}{5}$	1	5	0,44	0,06
A_5	$\frac{1}{5}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{5}$	1	0,24	0,04
Сума						6,73	1,00

Діаграма вагових коефіцієнтів для критеріїв A_1, A_2, A_3, A_4, A_5 представлена на рисунку 3.1.

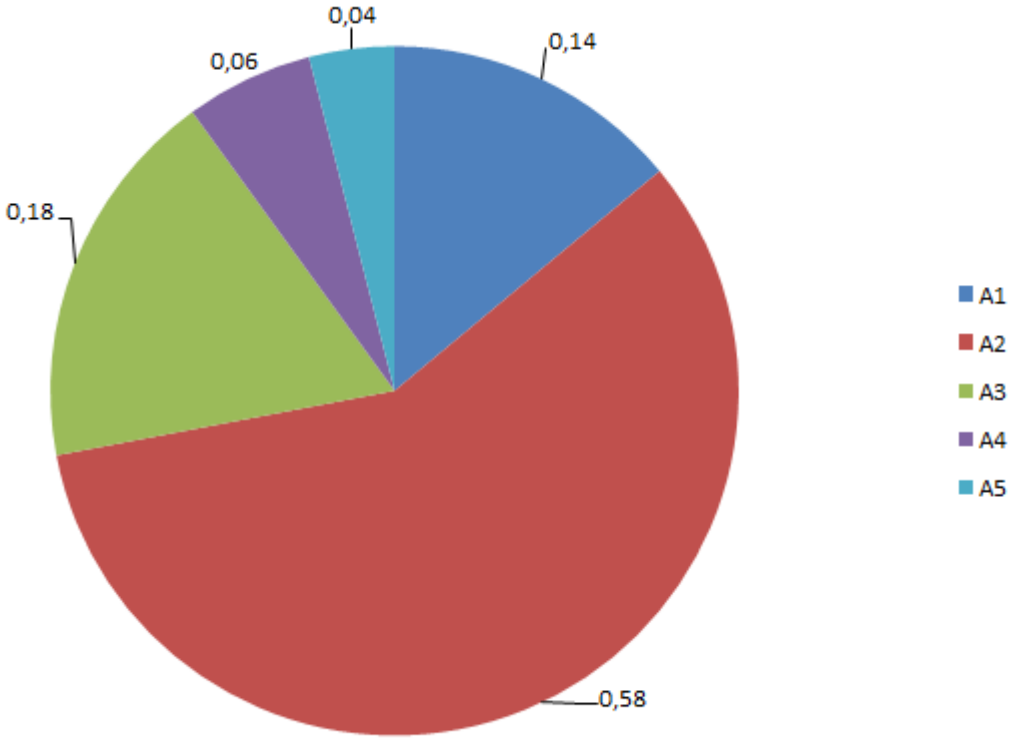


Рисунок 3.1 Вагові коефіцієнти критеріїв якості

Проведемо перевірку матриці попарних порівнянь несуперечливість.

Перевірка дозволяє виявити помилки, які міг допустити експерт при заповнені матриці парних порівнянь. Помилки (суперечливість) можуть бути наступними: наприклад, експерт вказав що критерій A_1 важливіший ніж критерій A_2, критерій A_2 важливіший ніж критерій A_3, і в той самий час критерій A_3 важливіший за критерій A_1. Розглянемо перевірку на несуперечливість для задачі виявлення ваги критеріїв:

1. Знаходимо суми стовбців матриці парних порівнянь (табл. 3.4):

$$R_j = \sum_{k=1}^{n} x_{kj} \tag{3.3}$$

Таблиця 3.4 Сума стовбців матриці парних порівнянь

Критерії	R_j
A_1	4,68
A_2	1,79
A_3	9,34
A_4	15,34
A_5	25

2. Розраховуємо допоміжну величину L шляхом додавання множення сум стовбців матриці на вагу критеріїв:

$$L = \sum_{i=1}^{n} k_i * M_i \tag{3.4}$$

Для даного прикладу $L = 5.23$.

3. Знаходимо величину, яка називається індексом узгодження (CI – Consistency Index):

$$CI = \frac{L - n}{n - 1} \tag{3.5}$$

Для даного прикладу $CI = 0.2$.

4. В залежності від розмірності матриці парних порівнянь знаходиться величина випадкового узгодження (RI – Random Index). Значення RI для матриць розмірності від 3 до 10 приведено в табл. 3.5.

Таблиця 3.5 Величини випадкового узгодження

Розмірність матриці	3	4	5	6	7	8	9	10
RI	0,58	0,90	1,12	1,24	1,32	1,41	1,45	1,49

В даному прикладі RI = 1,12.

5. Знайдемо відношення узгодженості (CR – Consistency Ratio):

$$CR = \frac{CI}{RI} \tag{3.6}$$

Якщо відношення узгодженості перевищує 0,2, то необхідне уточнення матриці парних порівнянь.

В даному прикладі $CR = 0.18$, отже, уточнення експертних оцінок в даному випадку не потрібне.

Використовуючи отримані коефіцієнти, визначемо інтегральний показник якості для алгоритмів:

1. Алгоритми пошуку асоціативних правил.
2. Алгоритми кластеризації (методом агломерації, ділення).
3. Алгоритми задач класифікації і регресії (в т.ч. прогнозування часових рядів).
4. Алгоритми побудови нейромереж і генетичні алгоритми.

Виберемо категоріальний шкалу від 0 до 7 (де 0 - якість не задовільно, 7 - гранично досяжний рівень якості на сучасному етапі) для функціональних можливостей програмних продуктів.

Значення вагових коефіцієнтів a_i відповідають функціональним можливостям продуктів:

1) аналіз слабоструктурованої інформації: a_2 = 0.4;
2) аналіз неструктурованою інформації: a_1 = 0.36;
3) класифікація документів за поданими категоріями: a_3 = 0.15;
4) генерація тематичної структури досліджуваного тексту: a_4 = 0.06;
5) вилучення інформації по конкретним об'єктам: a_5= 0.03.

Визначемо (по введеній шкалі) кількісні значення функціональних можливостей X_{ij} (таблиця 3.6). Визначимо інтегральний показник якості для кожного програмного продукту.

Таблиця 3.6 Інтегральний показник якості

Критерії	Вага	Алгоритми пошуку асоціативних правил	Алгоритми кластеризації	Алгоритми задач класифікації і регресії	Алгоритми побудови нейромереж і генетичні алгоритми	Базові значення
Точність розбиття на групи	0.14	6	6	4	6	4.4
Точність прогнозу	0.58	0	0	0	0	0
Точність находження закономірностей	0.18	3	3	5	7	3.6
Аналіз поганоформалізованих даних	0.06	0	0	0	7	1.4
Знаходження прихованих закономірностей даних	0.04	5	3	7	7	4.4
Інтегральний показник якості Q		1.54	1.47	1.69	2.77	1.49

де $Q_i = \sum a_i * X_{ij}$ – інтегральний показник якості для i -го програмного середовища.

Побудуємо пелюсткову діаграму інтегрального показника якості кожного програмного продукту (рис. 3.2).

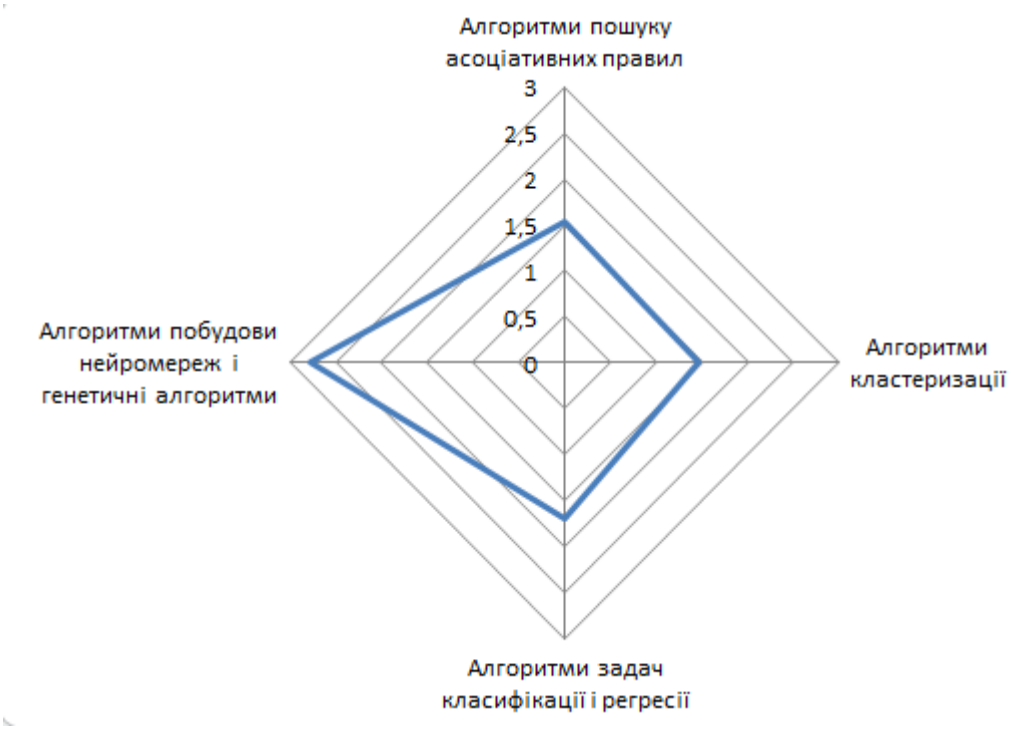


Рисунок 3.2 Пелюсткова діаграма інтегральних показників якості алгоритмів

Пелюсткова діаграма значень характеристик якості функціональних можливостей (критеріїв) представлена на рисунку 3.3.

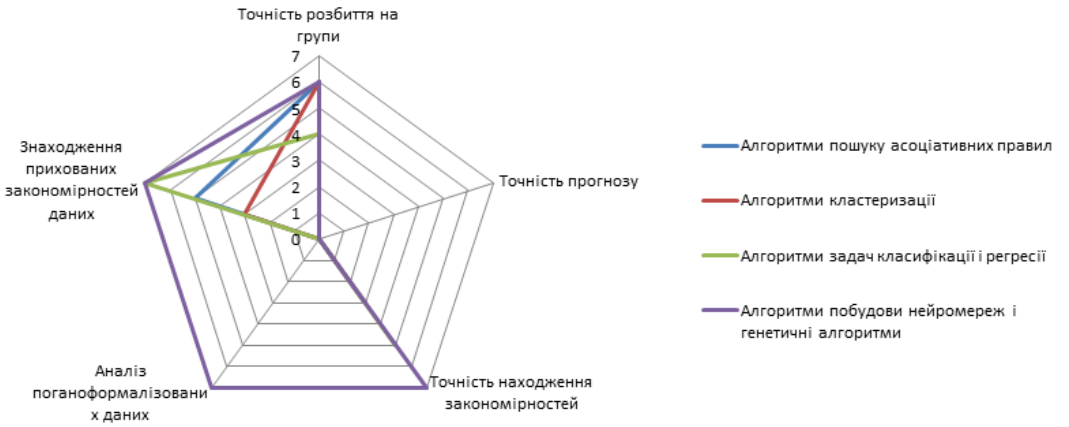


Рисунок 3.3 Пелюсткова діаграма значень функціональних характеристик

Висновки до розділу 3

В даному розділі ми скористались аналітичною ієрархічною процедури Сааті для порівняння алгоритмів аналізу неструктурованих та слабо структурованих даних.

Порівняльний аналіз алгоритмів аналізу неструктурованої і слабоструктурованої інформації показав, що три з чотирьох розглянутих алгоритмів мають значення інтегрального показника якості, що перевищує базове значення, - алгоритми пошуку асоціативних правил, алгоритми кластеризації, алгоритми побудови нейромереж і генетичні алгоритми. З розрахунків і графіка інтегральних показників якості програмних продуктів видно, що найбільший інтегральний показник якості має група алгоритмів побудови нейромереж та генетичних алгоритмів

РОЗДІЛ 4. КЛАСИФІКАЦІЯ ТЕКСТІВ З ВИКОРИСТАННЯМ IBM SPSS MODELER

Програмне забезпечення для інтелектуального аналізу допомагає знаходити неочевидні, приховані закономірності у великих наборах даних. У зв'язку зі стрімким зростанням обсягів текстової інформації класифікація текстів стала одним з ключових методів організації текстових даних. Класифікатори на базі методу опорних векторів (Support Vector Machines, SVM) з поданням текстових даних у вигляді набору слів (Bag-of-Words, BoW) демонструють високу ефективність класифікації в плані точності результатів і можливості узагальнення. Основними проблемами класифікації є витяг ознак і вибір параметрів SVM для досягнення найвищої ефективності. У статті пропонується загальна структура для створення класифікаторів текстів на базі IBM SPSS Modeler і Java без використання предметно-орієнтованих словників.

4.1 Постановка завдання і підхід до її вирішення

В даному розділі приведено формальне визначення задачі та загальний підхід до її вирішення. Є набір навчальних записів x_i, де i приймає значення від 1 до l (кількість навчальних записів). Кожний запис x_i є вектором ознак, який має розмірність k. З кожним записом x_i пов’язана мітка y_j, де j приймає значення від 1 до m (кількість визначених класів). Класифікація може розглядатися як функція відображення $y_j = f(x_i)$, яка кожному вхідному запису зіставляє один з визначених класів. Завдання створення класифікатора полягає в оцінці функції (класифікатора) f', яка забезпечує максимальну середню точність класифікації. Точність класифікації для кожного класу - це відношення кількості коректно класифікованих записів $\left(f'(x_i) = f(x_j)\right)$ до загальної кількості записів в класі. Середня точність обчислюється для всіх визначених класів.

Для класифікації текстів необхідно перетворити кожну вхідну текстову запис в вектор ознак, щоб отримати можливість побудувати класифікатор.

4.2 Архітектура та розробка системи

Для вирішення завдань інтелектуального аналізу даних і інтелектуального аналізу використовується стандартний процес CRISP-DM (Cross Industry Standard Process for Data Mining).

На рисунку 4.1 показана архітектура рішення, що включає три основних компоненти:

1. Попередня обробка даних - цей компонент забезпечує попередню обробку вхідних текстових даних для генерування векторів ознак на базі моделі Bag-of-Words.

2. Побудова і оцінка моделі - цей компонент забезпечує побудова моделі класифікації на базі згенерованих векторів ознак. Набір вхідних векторів ознак поділяється на піднабори training (навчання), test (тестування) та validation (перевірка). Піднабір training використовується для початкової оцінки SVM-моделі класифікації. Піднабір test використовується для більш точного налаштування початкових параметрів. Піднабір validation використовується для визначення ефективності моделі.

3. Сховище даних - це компонент забезпечує зберігання даних, що використовуються в рішенні, і управління ними. Є два основних типи даних - вхідні дані і згенеровані вектори ознак, а також допоміжні дані, які використовуються в компоненті попередньої обробки даних.

Проектування запропонованого рішення має дві цілі:

1. *Узагальнення.* Генерування і вибір ознак не залежить від будь-яких предметно орієнтованих словників. Кожен термін у документі розглядається як ознака, що використовується при класифікації. Крім того, для моделі класифікації обраний метод SVM, який забезпечує краще узагальнення, ніж інші методи класифікації, і

дозволяє уникнути перенавчання (overfitting). Це важливо при класифікації текстів, оскільки згенеровані вектори ознак зазвичай бувають розрідженими, що підвищує ймовірність перенавчання.

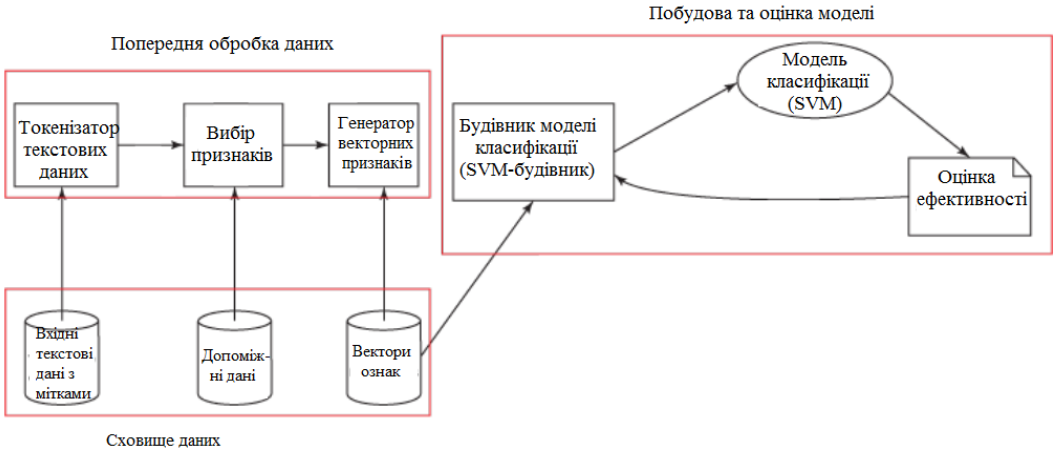


Рисунок 4.1 Архітектура рішення для класифікації ткстів на базі метода SVM

2. *Модульність.* Компоненти цього рішення можна легко модифікувати і / або замінювати. Наприклад, розробник може при необхідності додавати або видаляти фільтри при виборі ознак. Для побудови моделі класифікації розробник може замінити SVM на будь-який інший метод класифікації, який використовує модель Bag-of-Words, такий як k-NN (K-Nearest Neighbours, метод k найближчих сусідів) або дерева рішень.

4.2.1 Розробка рішення на базі процесу CRISP-DM

Як і в будь-яких інших проектах з інтелектуального аналізу даних, стандартним процесом для розробки є CRISP-DM.

Основні етапи процесу CRISP-DM:

• Розуміння завдання

• Розуміння даних

• Попередня обробка даних

• Моделювання

• Оцінка

• Розгортання

У представленому тут проекті етап попередньої обробки даних розробляється в середовищі Java, етапи моделювання та оцінки розробляються в SPSS Modeler, а репозиторій даних розробляється з використанням простих електронних таблиць Excel.

Метою проекту є створення системи класифікації нових текстових даних на основі визначеного набору класів. Ця система повинна працювати без будь-яких предметно-орієнтованих словників. Вхідними даними є набір забезпечених мітками текстових записів.

Далі наведені формальний опис моделі вхідних даних, опис функцій полів в створенні класифікатора і приклад очікуваних даних.

Вхідними даними є таблиця забезпечених мітками записів з наступними полями:

• *Ідентифікатор* - унікальний ідентифікатор кожного запису.

• *Текстові дані* - текстові дані, з яких беруться ознаки для створення класифікатора.

• *Клас запису* - використовується як мітка для кожного запису для створення класифікатора.

В таблиці 4.1 показана схема вхідних даних для будь-якого класифікатора текстів.

Таблиця 4.1 Загальна схема вхідних даних для будь-якого класифікатора текстів

Ідентифікатор	Текстові дані	Клас запису
L	T_1	y_1
D	T_2	y_j
B	T_3	y_m

В таблиці 4.2 приведений приклад вхідних даних о дефектах програмного забезпечення.

Таблиця 4.2 Приклад вхідних даних о дефектах програмного забезпечення

Ідентифікатор	Опис	R_Trigger
68971	See this in regression run can't reproduce it	l_coverage
68741	It’s not possible to build the BSF stream	l_build
68852	I built an ICM to parse a schema that contains optional dateTime attributes.	l_developer_test

Поле *Опис* в таблиці 4.2 відповідає полю *Текстові дані* в таблиці 4.1, а поле *R_Trigger*, яке представляє клас дефекту програмного забезпечення, описаний в цьому записі, відповідає полю *Клас записи* в таблиці 4.1.

Попередня обробка даних. Оскільки для створення класифікатора текстів планується використовувати метод Support Vector Machines, необхідна попередня обробка текстових даних, щоб перетворити їх в набір векторів ознак на базі уявлення Bag-ofWords. Іншими словами, кожний текстовий запис T_i буде перетворена в вектор $x_i = (x_1, x_2, \dots, x_{ij}, \dots, x_{ik})$, де x_{ij} - це витягнутий термін. Як буде показано далі, кожен витягнутий термін відповідає ознаці, який буде використовуватися для створення і виконання класифікатора документів. Етап попередньої обробки можна розділити на два кроки:

1. Вибір/скорочення ознак;
2. Генерування ознак.

Вибір/скорочення ознак. Ми створюємо компонент для вибору ознак як токенізатор термінів, за яким знаходиться набір фільтрів, що дозволяють отримати терміни, які є найбільш важливими в контексті класифікації. Схема модуля вибору ознак показана на рисунку 4.2, а потім наведено опис кожного елемента.

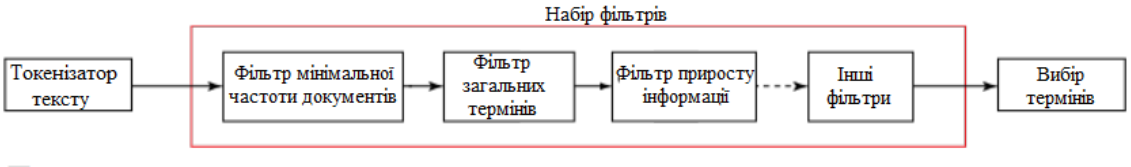


Рисунок 4.2 Схема модулю вибору ознак

Токенізатор термінів. Для вилучення всіх можливих термінів з текстових даних в нашому випадку використовується простий токенізатор, в якому роздільниками є пробіл. Для простоти пропонована реалізація буде грунтуватися на термінах, що складаються з одного слова. Але після застосування токенізатора до реального документу виникає проблема отримання великих наборів ознак (що включають тисячі елементів) для кожного документа. В контексті методів класифікації ця проблема називається «прокляттям розмірності» (curse-of-dimensionality). Прокляттям - тому що в міру збільшення кількості ознак (нагадаємо, що кожен витягнутий з документа термін відповідає ознакою) час побудови моделі зростає експоненціально. Тому необхідні ефективні методи вибору з вхідних даних найбільш значущих термінів без істотної втрати інформації і подальшого негативного впливу на точність класифікації. Ось чому для створення класифікатора необхідний вибір ознак.

Фільтр мінімальної частоти документів. Щоб переконатися в тому, що термін часто використовується в навчальних даних, і щоб виключити великі, непотрібні вектори ознак, цей фільтр видаляє терміни, частота появи яких в документах нижче встановленого порогового значення.

Фільтр загальних термінів. Будь-який текст містить загальні терміни, які не є предметно-орієнтованими і мають низьку дискримінаційну здатність для розподілу записів за категоріями. Ми використовуємо список з 5000 найбільш вживані слів та/або лем, взятих з найбільшого і збалансованого корпусу американської англійської мови Corpus of Contemporary American English, що включає 450 мільйонів слів.

Одним з варіантів застосування цього фільтра є безпосередній пошук загальних термінів у списку витягнутих термінів і їх видалення. Однак такий підхід має дві серйозні проблеми.

• Вхідні дані можуть бути не чистими. Наприклад, слово enough (досить) вважається загальним ключовим словом, але в зв'язку з помилками або іншою причиною виникнення помилок в даних воно може виглядати як enoug, enogh, enougth і т. Д. Такі слова пройдуть через фільтр, що небажано.

• Загальні слова зберігаються в вихідному репозиторії як леми і можуть з'являтися у вхідних текстових даних як морфологічні варіанти. Наприклад, слово produce (виробляти) є загальним терміном, але може з'являтися в формах produced, product, producing і т. Д. Безпосередній пошук не дозволяє видаляти такі варіанти.

Інтуїтивне рішення для цих двох проблем полягає в тому, щоб видаляти з тексту будь-яке слово, яке схоже якомусь слову зі списку загальних термінів, а не просто збігається з ним. Для реалізації цього підходу необхідно спочатку визначити міру схожості двох слів. В даному випадку для оцінки схожості кожного витягнутого терміна з елементами в списку загальних слів використовується нормалізована міра схожості Левенштейна. Фільтри загальних термінів відкидають термін, якщо його показник схожості з будь-яким загальним терміном перевищує задане граничне значення. Після виконання цього фільтра більшість загальних термінів, що з'являються як леми або морфологічні варіанти, або навіть введених з друкарськими помилками, повинні бути видалені. В Додатку А описано обчислення фільтра Левендштайна.

Фільтр приросту інформації. Після видалення загальних слів необхідно витягти терміни, які найбільш корисні для класифікації текстів. Корисність означає найвищу дискримінаційну здатність для віднесення термінів документів до правильного класу. Для оцінки дискримінаційної здатності в даному випадку використовується приріст інформації (information gain, IG) кожного терміна, або ентропія. IG-фільтр працює наступним чином: для кожного терміна, що надходить з попередніх фільтрів, обчислюється IG, а потім вибирається співвідношення з термінів, які мають найвищі значення IG.

Тут ми кажесо, як обчислюється приріст інформації для кожного терміну. Нехай P_t - глобальна ймовірність класу i, де $1 \leq i \leq k$, а $p(t)$ вірогідність класу i, якщо документ містить термін t. Нехай $F(t)$- частка документів, що містять термін t. Тоді приріст інформації $IG(t)$ будь-якого терміна t, визначається наступним чином:

$$IG(t) = -\sum_{i=1}^{k} P_i \log(P_i) + F(t) \sum_{i=1}^{k} p_i(t) \log\big(p_i(t)\big) + (1 - F(t)) \sum_{i=1}^{k} \big(1 - p_i(t)\big) \log(1 - p_i(t))$$

Реалізація фільтру приросту інформації в динаміній пові програмування представлено у Додатку Б.

Фільтр загальних слів поміщений перед фільтром приросту інформації з наступних причин:

• Загальні терміни визначаються як слова, які найбільш часто зустрічаються в більшості документів англійською мовою, тому зрозуміло, що їх коефіцієнт приросту інформації в будь-якому випадку буде низьким.

• Складність обчислень визначається як O(l.d.k), де l - це кількість записів в навчальних даних, d - це кількість термінів (розмірність векторів ознак), k - це кількість класів. Застосовуючи фільтр загальних термінів перед IG-фільтром, можна значно зменшити значення d, що скорочує час виконання IG-фільтра.

Генерування ознак. Тепер, після вибору піднабору термінів, найбільш корисних в контексті класифікації, для кожного запису є набір ознак

(нагадаємо, що кожен термін розглядається як ознака). Наступним кроком є обчислення важливості кожної ознаки в кожному документі. У нашому випадку показником важливості ознаки є добре відома статистична міра Term Frequency-Inverse Document Frequency (tf-idf). Оскільки для кожного запису є вектор ознак, така сукупна структура даних іноді називається матрицею ознак (матриця - це група векторів).

$$tf - idf(t, D) = tf(t, D) \times idf(t)$$

де

tf(t,D): документ частоти терміну *t* у документі *D*, оскільки *tf* збільшується, це означає, що цей термін тісно пов'язаний з документом *D*. Однак, якщо цей термін також часто згадується в інших документах, це зменшує співвідношення між терміном *t* та документом *D*, тому *tf* переважаэ *idf* як буде пояснено нижче.

Idf(t): є частотою зворотного документа терміна *t*, яка визначається як $idf(t) = \log(\frac{N}{df(t)})$, де *N* - кількість документів/записів у наборі навчальних даних, *df(t)* - це число документа, де *t* згадується (документ частоти терміну *t*). Оскільки простір терміна *t* збільшується, це означає, що термін є рідким і не тісно пов'язаний з усіма документами.

Після обчислення показників *tf-idf* для всіх термінів/ознак в навчальних даних генерується матриця ознак, що має наступну структуру:

- Кожен рядок представляє запис - в нашому випадку кожен запис є документом з унікальним ідентифікатором.
- Кожен стовпець представляє унікальну ознаку/термін.
- Кожна клітинка в матриці має значення *tf-idf* для конкретного терміна (відповідного стовпчика) в конкретному документі (відповідному рядку).

4.3 Моделювання

Після перетворення даних в модель Bag-of-Words можна без втрати спільності переформулювати розглянуту задачу в наступну задачу бінарної класифікації: є набір записів з векторами ознак x^i, які необхідно розподілити по класам $x^i = \{-1,1\}$. В SVM потрібно знайти гіперплоскість, яка найкращим чином розділяє елементи даних на два класи.

На рисунку 4.4 показаний потік SPSS Modeler для побудови та оцінки SVM-моделі, призначеної для класифікації текстових документів. Потік включає вузли для читання векторів ознак, поділу вхідних записів, побудови моделі і оцінки моделі.

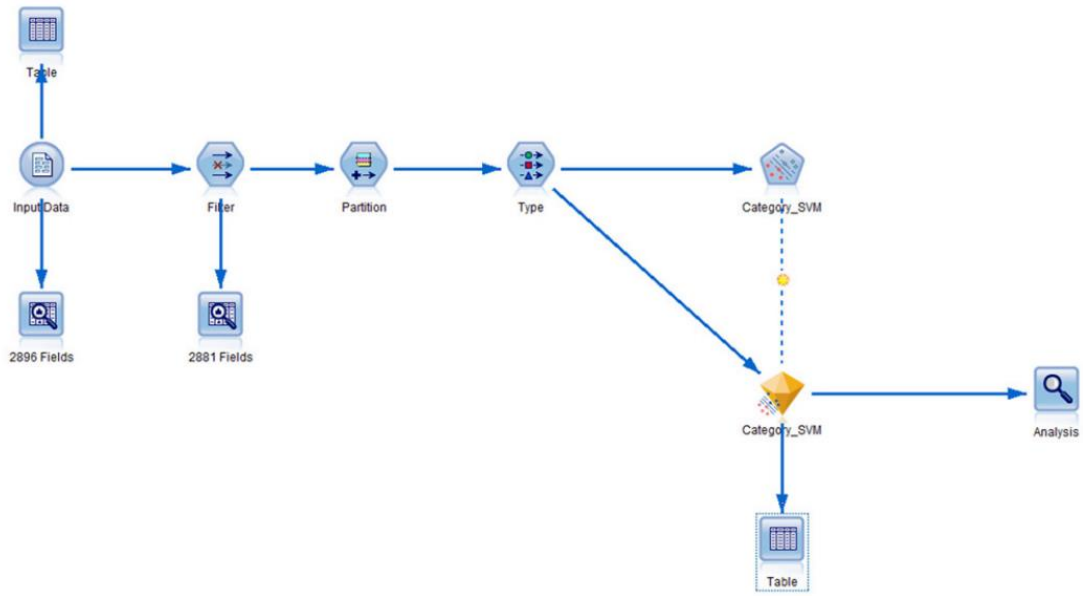


Рисунок 4.4 Потік SPSS Modeler для створення SVM-класифікатора

Вузол *Input Data* призначений для читання CSV-файлу, який містить таблицю векторів ознак, що згенерувала з використанням модуля попередньої обробки даних. Вузол *Filter* використовується для видалення будь-яких непотрібних полів, таких як ідентифікатор запису, який зазвичай забирається, оскільки не використовується в моделюванні. Крім того, в прикладі з даними про дефекти після застосування фільтрації деякі маркери

з'являються, але вони не мають ніякого значення в предметній області, тому видаляються. Зверніть увагу, що в вузлах *Data Audit* фільтрація скоротила кількість полів з 2896 до 2881.

Вузол *Partition* розділяє дані на три групи - training, test і validation (навчання, тестування і перевірка). При створенні будь-якої моделі класифікації рекомендується розділяти вхідні дані на такі групи. Піднабір training використовується для побудови моделі з заданими параметрами і оцінки (початкової) її внутрішніх коефіцієнтів. Піднабір test використовується для тестування точності класифікації з метою налаштування параметрів навчання моделі, її повторного побудови та оновлення її внутрішніх коефіцієнтів для підвищення точності. Піднабір validation використовується для оцінки точності класифікації нових даних. Поднабор validation не впливає на процес побудови моделі безпосередньо (як дані для навчання) або опосередковано (як дані для тестування), тому визначення точності на базі поднабора validation є хорошим методом оцінки ефективності застосування побудованого класифікатора до нових непомічених даних.

Вузол *Type* вказує роль кожного поля. Всі поля з витягнутими ознаками позначаються як *Input* (вхідні дані) для навчання класифікатора, а поле *Category* відзначається як *Target* (результат) для використання в якості поля класу для кожного запису.

Category_SVM - це вузол побудови SVM-моделі з використанням витягнутих ознак як вхідних даних і поля *Category* як результату (відповідно до конфігурації вузла Type). Ми використовуємо режим *Expert* для отримання більшого контролю при налаштуванні параметрів класифікатора. Основною проблемою буде вибір кращого набору параметрів для отримання максимально можливої точності.

4.4 Оцінка

Вузол *Analysis* в потоці SPSS Modeler використовується для оцінки ефективності моделі шляхом порівняння передбаченого значення класу з вихідним класом для кожного піднабору (training, test і validation). Він показує середню точність моделі при її застосуванні до кожного піднабору. Точність класифікації для кожної категорії обчислюється як відношення кількості коректно класифікованих записів до загальної кількості записів в цій категорії. Середня точність обчислюється для всіх категорій.

На рисунку 4.5 показано конфігурація вузла *Analysis* для відображення середньої точності класифікації для всіх категорій, згрупованої для кожного піднабору. Зверніть увагу, що в поле Separate by partition поставлена позначка, щоб розділяти результати визначення точності по піднабору training, test і validation. Це дозволить вибрати найкращі значення параметрів і уникнути проблеми перенавчання. Перенавчання виникає в тому випадку, коли точність є дуже високою на піднаборі для навчання і тестування і дуже низькою на піднаборі для перевірки. Це означає, що модель стає прив'язаною до даних для навчання і тестування і не є достатньо загальної для класифікації будь-яких нових вхідних даних.

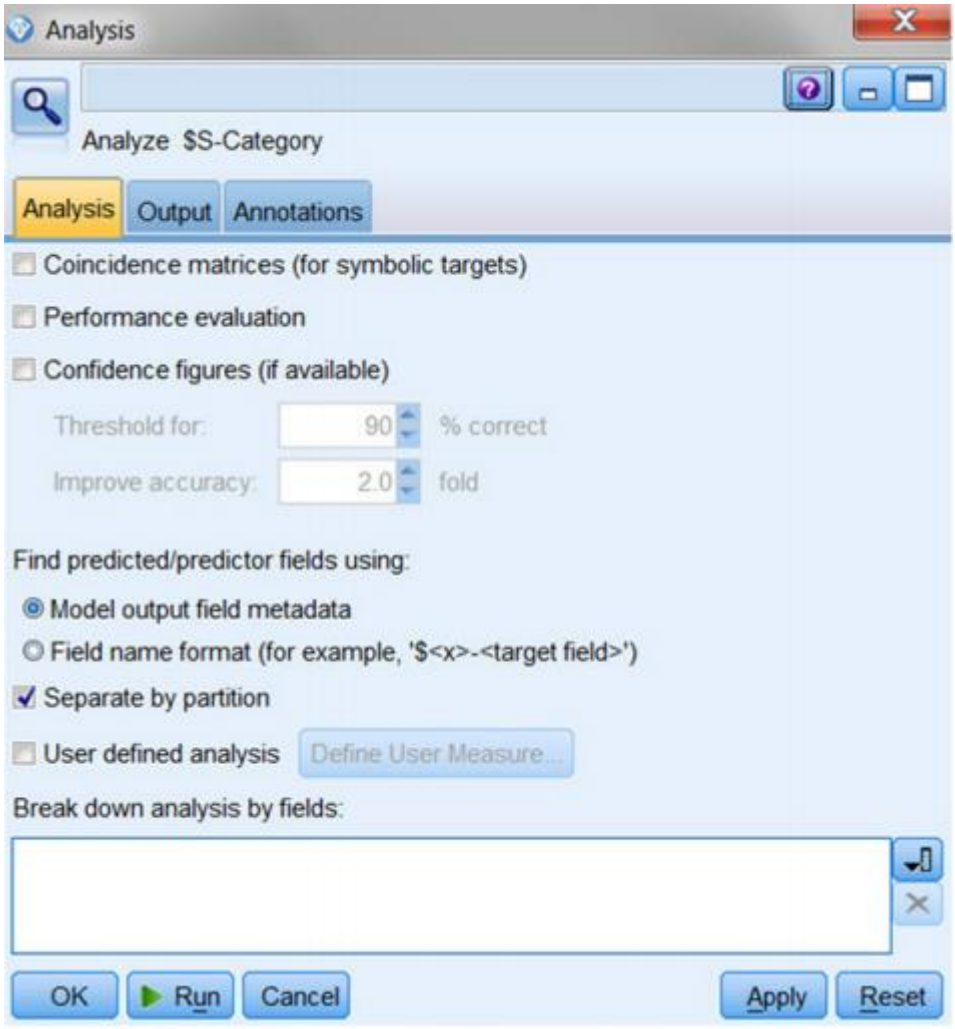


Рисунок 4.5 Конфігурація вузла Analysis для відображення середньої точності, згрупованим по піднаборам

Розгортання. Розгорнемо наше рішення як інтеграцію процесів між Java-програмою для попередньої обробки даних і потоком побудови моделі з SPSS Modeler. Компоненти для попередньої обробки даних розгортаються як окрема Java-програма, що генерує файл векторів ознак в форматі CSV (Comma Separated Values), потім цей CSV-файл використовується як вхідні дані в потоці SVM-класифікатор.

Згенерований файл векторів ознак відправляється в потік SPSS Modeler для побудови SVM-моделі. Модель будується з використанням створеного нами потоку наступним чином:

1. В SPSS Modeler відкрити потік SVM_Stream.str.

2. Вибираємо потрібний файл векторів при знаків для вузла *Input Data*, як показано на рисунку 4.6.

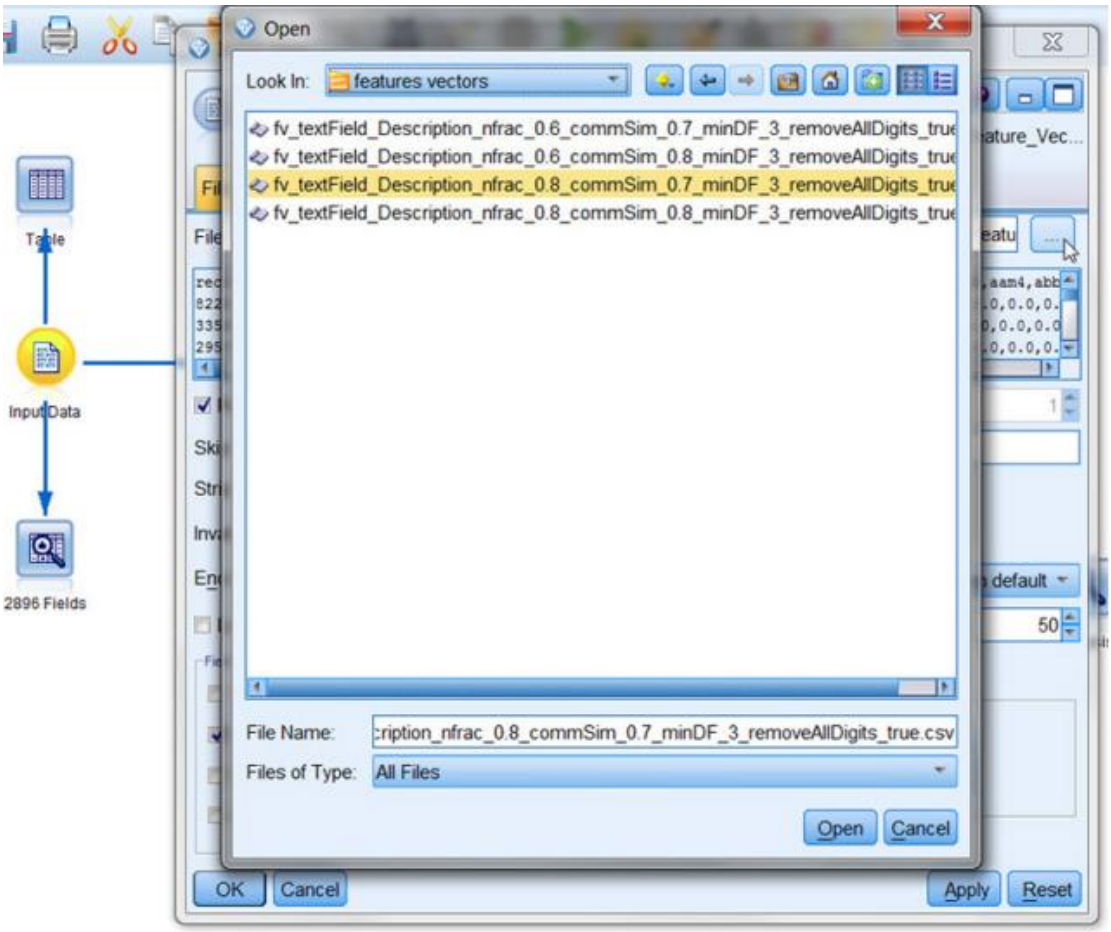


Рисунок 4.6 Вибір файла векторів ознак в якості вхідних даних для побудови SVM-моделі в SPSS Modeler

3. Запустимо потік кнопкою *Run***.** Буде згенеровано зліпок моделі з іменем *Category_SVM* (рис. 4.7).

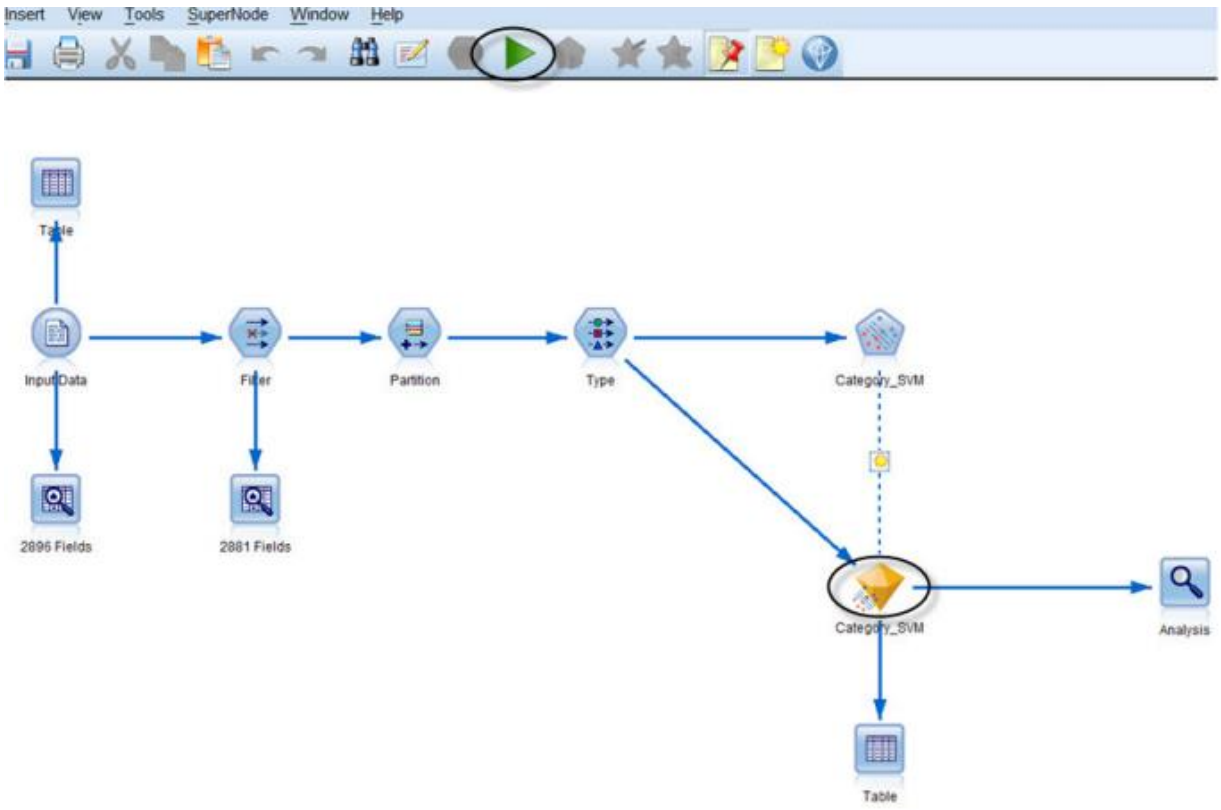


Рисунок 4.7 Запуск створення SVM-моделі

4.5 Налаштування експерименту і вибір параметрів

Описана система має параметри, які можна розділити на дві основні групи:

- параметри генерування і вибору ознак;
- параметри побудови SVM-моделі.

4.5.1 Налаштування параметрів генерування і вибору ознак

Ці параметри конфигурируются в файлі config.properties. Нижче наведено перелік параметрів з описами.

- *inputDataFileName* - шлях до вхідного файлу .XLS, який містить текстові записи для створення класифікатора;

- *commonTermsListFilePath* - шлях до файлу .XLS, який містить список загальних ключових слів для етапу фільтрації в модулі вибору ознак;
- *minTermDocFreqThr* - порогове значення мінімальної частоти документів для кожного терміна для проходження через фільтр;
- *logFilePath* - шлях файлів журналів;
- *idFieldColHeader* - заголовок стовпця, що містить ідентифікатори записів;
- *textFieldColHeader* - заголовок стовпця, що містить текстові дані, які використовуються для створення класифікатора;
- *categoryFieldColHeader* - заголовок стовпця, що містить класи для класифікації (в нашому прикладі це категорія кожного запису, що представляє дефект програмного забезпечення);
- *commonKeywordSimilarityThresholdVals* - список значень, розділених комами, для фільтра видалення загальних термінів; при зменшенні порогового значення фільтр стає більш консервативним і видаляє більше термінів, які виглядають загальними;
- *splitter* - символ, який використовується для поділу значень у файлі конфігурації (в нашому прикладі кома);
- *fractionTopNumTermsIGvals* - список значень частки обраних термінів для найбільшого приросту інформації;
- *commonKeywordSimilarityThresholdVals* - список значень параметра порогового значення для фільтра загальних термінів;
- *minTermDocFreqThr* - параметр для фільтра частоти документів;
- *removeAllDigitsTerms* - прапор для визначення потреби у видаленні термінів, що містять тільки цифри;
- *outFeaturesVectorsDir* - вихідний каталог для генерування векторів ознак.

4.5.2 Налаштування параметрів побудови SVM-моделі

Для побудови SVM-моделі з використанням згенерованих векторів ознак зконфігуріруємо набір параметрів в режимі *Expert*, як показано на рисунку 4.8 і описано далі.

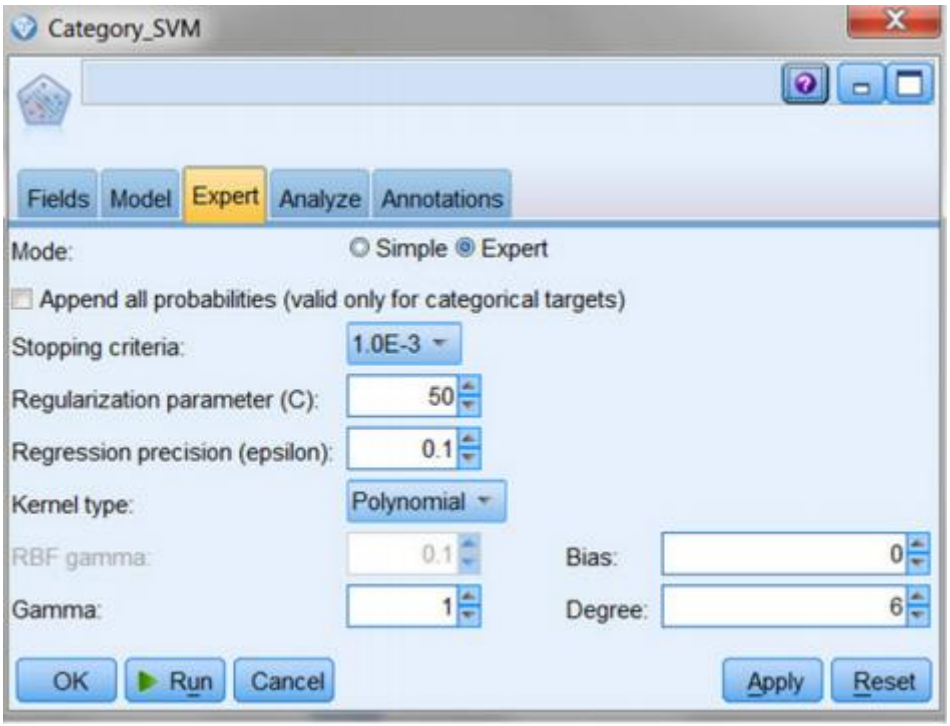


Рисунок 4.8 Параметри для побудови SVM-моделі з використанням вузла SPSS

Важливими параметрами для побудови SVM-моделі є наступні:

- *Stopping criteria (Критерії зупинки).* Це різниця між розрахунковими внутрішніми коефіцієнтами SVM-моделі після кожної ітерації оптимізації в SVM з використанням алгоритму Sequential Minimal Optimization (SMO). Збільшення цього параметра буде підвищувати точність, збільшуючи час побудови моделі.

- *Regularization parameter (C)* (Параметр регуляризації). Цей параметр вказує вагу для суми помилок в SVM в цільової функції для оцінки оптимальної гіперплощини в SVM. Він встановлює баланс між помилкою на навчальному наборі (в межах вибірки) і відступом. Наприклад, при збільшенні параметра C метод SVM визначає гіперплоскость таким чином, що елементи навчання стають ближче до кордонів класифікації і відступ, що розділяє зменшується, що може призводити до неправильної класифікації нових елементів поблизу кордонів (тобто до перенавчання). При зменшенні параметра C метод SVM стає більш консервативним, оскільки відступ збільшується і помилкам класифікації навчальних елементів присвоюється менша вага. Іншими словами, збільшення C підвищує точність на навчальних даних, але може привести до перенавчання.
- *Kernel type (Тип ядра).* Функція ядра використовується для нелінійного зіставлення конкретного вектора ознак з простором вищої розмірності, при якому підвищується лінійна подільність даних при збільшенні витрат на обчислення. Типовою функцією ядра є поліноміальна ступінь d радіально базисна функція від гамма радіусу.

Вибір параметрів. При виборі параметрів ми використовуємо простий підхід на базі полурешітчатого пошуку для визначення найкращого набору параметрів. Як уже згадувалося, критерієм оцінки для виконання решітчастого пошуку є загальна середня точність на піднаборі даних для тестування - тобто метод побудови моделі оцінює початкові внутрішні коефіцієнти моделі з використанням піднабору для навчання, перевіряє і настроює параметри моделі на базі піднабору для тестування. Цей підхід призводить до кращої точності і виключає проблеми перенавчання.

Ми припускаємо, що параметри (C, Kernel function і Stopping criteria) не корелюють, тому використовуємо наступний метод пошуку параметрів:

1. Першочергові налаштування параметрів (за замовчуванням в SPSS Modeler):
    - C=10;
    - Kernel function = Radial;
    - Gamma = 1;
    - Stopping criteria = 10^{-3}.
2. Пробуємо значення параметра C від 0,1 до 1 з кроком 0,1 і від 10 до 100 з кроком 10, поки не буде знайдено значення з найкращо. точністю на піднаборі навчальних даних.
3. Для значення C, знайденого на кроці 2, пробуємо типи ядра Radial, Linear і Polynomial, поки не буде знайдено варіант, що забезпечує найкращу точність на наборі навчальних даних
4. Для ядра, знайденого на кроці 3, пробуємо наступні діапазони параметрів, поки не буде отримана найкраща точність:
    - Якщо Kernel Function = Radial, пробуємо Gamma від 0,1 до 5 з кроком 0,1.
    - Якщо Kernel Function = Polynomial, пробуємо Degree від 1 до 10 з кроком 1.
5. Для критеріїв зупинки пробуємо від 10^{-3} до 10^{-6}, домножуючи на 10^{-1}.

Виконавши дії 2-5, ми отримали набір параметрів для побудови SVM-моделі. Було виявлено, що найкращу точність забезпечує набір параметрів, що наведені у таблиці 4.3.

Таблиця 4.3 Оптимальна конфігурація набору параметрів для досягнення найкращої середньої точності

Етап	Ім’я параметра	Значення параметра
Вибір ознак	minTermDocFreqThr	3
	commonKeywordSimilarityThresholdVals	0,7
	fractionTopNumTermsIGvals	0,8
	removeAllDigits	True
Побудова моделі	C	50
	Kernel Type	Polynomial
	Kernel Parameter	Polynomial Degree=6
	Stopping criteria	10^{-3}

Вихідний набір даних для навчання включає 4783 записів і одне текстове поле (Description). Згенерований вектор ознак містить таку ж кількість записів і 2896 полів. Побудова SVM-моделі виконувалося з використанням системи на базі двоядерного процесора Intel з тактовою частотою 2,7 ГГц і 1 ГБ оперативної пам'яті, зарезервованої для серверного процесу IBM SPSS Modeler Version 16. Для побудови моделі було потрібно 2 хвилини 10 секунд. У таблиці 4.4 показані результати точності, отримані з вузла Analysis, в порівнянні з вихідними даними, з передбачуваною категорією для кожного запису.

Таблиця 4.4 Загальна точність запропонованої системи, отримана з вузла Analysis

Результати в вихідному полі Category в порівнянні з $S-Category						
Піднабір	1_Training		2_Testing		3_Validation	
Коректно	2247	67,48%	516	65,16%	306	64,42%
Невірно	1083	32,52%	462	34,84%	169	35,58%
Всього	3330		978		475	

4.6 Висновки до розділу 4

У четвертому розділі був описаний інструментарій для практичної реалізації класифікатору тексту в середовищі моделювання SPSS Modeler.

Була поставлена задача і підхід до її вирішення, показана архітектура рішення, було розроблене рішення на основі процесу CRISP-DM, проведено моделювання, оцінку та аналіз створенного класифікатора текстів.

У четвертому розділі описано метод пошуку наборів параметрів для забезпечення найкращої точності і вказана таблиця з оптимальною конфігурацією набору параметрів для досягнення найкращої середньої точності.

ВИСНОВКИ

У даній роботі був розроблеми класифікатор тексту на базі процесу CRISP-DM у середовищі SPSS Modeler. Були розібрані існуючі алгоритми аналізу неструктурованої інформації та за допомогою аналітичної ієрархічної процедури Сааті проведено порівняльний аналіз алгоритмів.

В даному досліджені було використано системний підхід щодо досягнення цілей роботи, міждисциплінарний підхід, а також методи логічного аналізу і синтезу.

Відповідно до поставленої мети були виконані наступні завдання:

– розглянуто основні алгоритми аналізу неструктурованої та слабоструктурованої інформації;
– розроблено критерії оцінки алгоритмів неструктурованої та слабоструктурованої інформації;
– розроблено класифікатор тексту на основі процесу CRISP-DM;
– реалізовано класифікатор тексту у середовищі моделювання SPSS Modeler;
– апробовано роботу моделі.

Прототип був реалізований у вигляді моделі у ПЗ SPSS Modeler, також було описано усі кроки по реалізації моделі. Реалізована модель добре показала себе.

Розроблена модель залишає великі можливості щодо його поліпшення та модифікації. наприклад:

- автоматизація вхідного потоку;

- тестування моделі з іншими методами класифікації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Черняк Л. «Большие данные» – новая теория и практика / Л. Черняк. // Открытые системы. – 2011. – №10.

2. Отчет AAPOR о «Больших данных» [Електронний ресурс] // Американская ассоциация исследователей общественного мнения. – 2015. – Режим доступу до ресурсу: https://wciom.ru/fileadmin/file/nauka/grusha2015/AAPOR_big_data.pdf.

3. Корытникова Н.В. Online Big Data как источник аналитической информации в online-исследованиях. // Социс. – 2015. – №8. – С. 14–24.

4. Григорьев А.А., к.т.н., доцент кафедры «Информационных систем в экономике и менеджменте» РЭУ им.Г.В.Плеханова; Антоненко В.Д., к.э.н., доцент кафедры Статистики РЭУ им.Г.В.Плеханова , статья «Алгоритмы кластеризации», 34 с.

5. Суслов С. А., к. э. н., доцент кафедры «Экономика и статистика» НГИЭИ. Статья «Кластерный анализ: сущность, преимущества, недостатки»

6. Котов А., Красильников Н. Статья «Кластеризация данных». 2006. -16 с.

7. Луценко Е.В., д. э. н., к. т. н., профессор кафедры АСОИУ физического факультета АГУ, Коржаков В.Е., к. т. н., зав. кафедрой АСОИУ физического факультета АГУ. Статья «Некоторые проблемы классического кластерного анализа», 12 с.

8. Рыбанов А. Определение весовых коэффициентов сложности учебного курса на основе алгоритма Саати / А. Рыбанов. // Педагогические измерения. – 2014. – №4.

9. Лакаев А.С. Разработка интеллектуальных технологий и методов обработки неструктурированной информации // NovaInfo.Ru. 2013. Т. 1. № 27.

10. IBM RedBooks. Building Big Data and Analytics Solutions in the Cloud [Електронний ресурс] Режим доступу: http://www.redbooks.ibm.com/abstracts/redp5085.html?Open.

11. IBM RedBooks. Unlock Big Value in Big Data with Analytics [Електронний ресурс] Режим доступу: http://www.redbooks.ibm.com/abstracts/redp5026.html?Open.

12. IBM RedBooks. Big Data Networked Storage Solution for Hadoop [Електронний ресурс] Режим доступу: http://www.redbooks.ibm.com/abstracts/redp5010.html?Open.

13. IBM RedBooks. Analytics in a Big Data Environment [Електронний ресурс] Режим доступу: http://www.redbooks.ibm.com/abstracts/redp4877.html?Open.

14. Foreman J. Data Smart: Using Data Science to Transform Information into Insight / John W. Foreman., 2014. – 409 с.

15. Фридман Д. The Elements of Statistical Learning / Д. Фридман, Т. Хэсти, Р. Тибширани., 2003. – 552 с.

16. Kumar V. Introduction to Data Mining / V. Kumar, M. Steinbach, P. Tan. – Harlow: Pearson Education, Inc., 2006. – 736 с.

17. Бабич М.В. Алгоритм аналізу неструктурованої та слабо структурованої інформації [Електронний Ресурс] / Бабич М.В. // Міжнародна науково-практична конференція «Наука та освіта: ключові питання сучасності»: зб. наук. праць «λόгοσ». – 2018. - Режим доступу: http://ukrlogos.in.ua/17.10.2015_74.pdf.

ДОДАТОК А
Опис фільтру Левенштайна

Initialization:

$D(i, 1) = i$;

$D(1, j) = j$;

Recurrence Relation:

for each $i = 1 \ldots M$

for each $j = 1 \ldots N$

$$D(s[i], t[j]) = min \begin{cases} D(s[i-1], t[j]) + 1 // \text{вставка} \\ D(s[i], t[j-1]) + 1 // \text{видалення} \\ D(s[i-1], t[j-1]) + \delta(s[i], t[j]) // \delta(s[i], t[j]) = \begin{cases} 0: s[i] = t[j] \\ 1: s[i] \neq t[j] \end{cases} \end{cases}$$

end

end

Termination:

$LevenDist = D(s[M], t[N])$.

Реалізація фільтру Левенштайна з використанням динамічного програмування

Input

commonTermsList: Список загальних ключевих слів

ExtractedTermsList: Список термінів, що передаються з токенізатору

SimThreshold : Параметри фільтру

Process

forall *w* ***in*** *commonWordList*

forall *t* ***in*** *ExtractedTermsList*

if(*LevenSim(t,w)* > *SimThreshold*) //термін «приблизно» є загальним словом

remove *t* ***from*** *ExtractedTermsList*

endif

end

end

return *ExtractedTermsList*

ДОДАТОК Б

Реалізація фільтру приросту інформації з використанням динамічного програмування

Input

ExtractedTermsList: Перелік термінів, що передається токенізатором

TopIGfrac: Частина термінів, вибраних на основі IG, коливається від0 до1

Process

*numTopIGTerms=**Ceiling**(TopIGfrac***size**(ExtractedTermsList))*

*sortedIGTermList=**sort**(ExtractedTermsList in descending order based on IG)*

***for**(i=1 **to** numTopIGTerms)*

*   **put** ExtractedTermsList [i] **into** filteredTermsList*

end-for

***return** filteredTermsList*